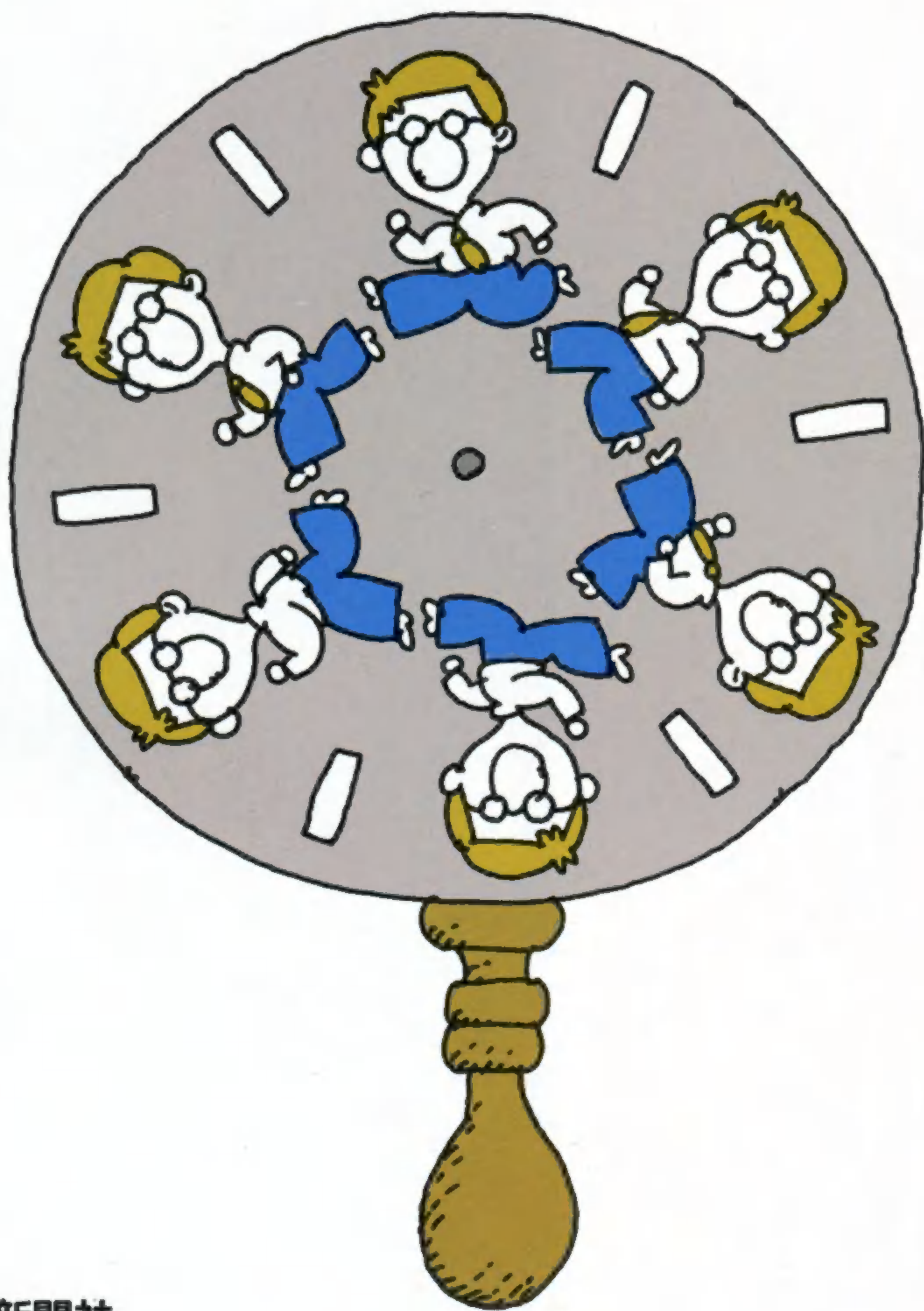


Quick BASIC

プログラミング500題

田中 廣——著



日刊工業新聞社

Quick BASIC プログラミング500題

田中 廣 著



日刊工業新聞社

Quick BASIC プログラミング500題

田中 廣——著



日刊工業新聞社

はじめに

“コンピュータ言語として何がよいか”という議論があります。とくに“パソコン用として何がよいか”となるとユーザー層がひろいだけに、一層多くの議論があります。最近では、パソコン用言語としては、“BASIC”対“C言語”の対比という形で一般化されています。そのおのこのの長所、欠点はいろいろなところでとりあげられていますので省略しますが、一言でいえば“BASIC”のフレンドリー（親しみやすい、わかりやすい）な面はすばらしいと思います。

“Quick BASIC”はフレンドリーを残しながら、従来の BASIC の欠点といわれてきた、①構造化、②モジュール化を可能にしました。インタープリタ形式のように、プログラムの1行ごとの入力チェックを行いながら、全体をコンパイルするコンパイル形にしています。コンパイル形の場合、コンパイル、リンクと作業が複雑になるのですが、“統合環境”と呼ぶエディット（プログラムの作成）→コンパイル（オブジェクトファイルの作成）→リンク（実行可能ファイルの作成）を簡単に行える環境を準備しています。加えて、デバッガによるプログラムのチェックも簡単にできるようになっています。つまり、“BASIC”のよさを残しながら“C言語”のよさをとりこんでいるわけです。したがって、入門者から一般のユーザーあるいはプログラム開発の専門家まで、“BASIC”をもう一度見直し、十分に使っていけるような言語になっています。

本書は「Ⅰ 準備編」で、“Quick BASIC”のバックアップコピーのとり方、セットアップの仕方から、ドロップダウンメニューの操作・機能を説明しています。Ⅱ編以降と並行して使ってください。「Ⅱ プログラム編」、「Ⅲ グラフィック編」はおのこの“Quick BASIC”を使ったプログラムのつくり方です。「Ⅳ 応用編」では従来の N-BASIC との関連、再帰プログラム等を紹介しています。例題 272 題、演習問題 211 題、応用問題 20 題による、“プログラミング 500 題”シリーズの一環としています。

なお、“Quick BASIC” Ver. 4.5（音楽等の機能が追加されています。Ⅱ編 26 章参照）が最新版ですが Ver. 4.2 でも使えるように工夫してあります。

確認用に使用した機器は PC-9801 VM です。PC-9800 シリーズで使えますが、一部の機種の高解像モード（XA, LT, XL, XL 2, RL）では使えません。OS は MS-DOS を使っています。MS-DOS は現在 Ver. 3.3 が最新ですが 2.0 以上で使用できます。

1989 年 11 月

田中 廣

目 次

I 準備・環境コマンド編

1 章	準 備..... 1
	[例題 1] ~ [例題 9] 1
2 章	プログラムの作成・ランタイム実行・保存・ 終了..... 23
	[例題10] ~ [例題16] 23
3 章	プログラムの読み込み・修正・保存・サブ ルーチンとファクションの作成・印刷..... 31
	[例題17] ~ [例題24] 31
4 章	ダイレクト実行..... 39
	[例題25] ~ [例題28] 39
5 章	サブファイル・テキストファイル・メイン プログラムの設定・インクルードファイル..... 43
	[例題29] ~ [例題38] 43
6 章	表 示..... 53
	[例題39] ~ [例題43] 53
7 章	編 集..... 59
	[例題44] ~ [例題47] 59
8 章	検 索..... 63
	[例題48] ~ [例題52] 63
9 章	デバッグ・関数..... 68
	[例題53] ~ [例題66] 68

Ⅱ プログラム編

1 章	変数の型・四則演算・累乗・整数除算・剰余..... 82 %, !, \$, &, #, +, -, *, /, ^, ¥, MOD, DEF INT/SNG/LNG/DBL [例題 1] ~ [例題10] 82 [演習問題] (1)~(13) 88
2 章	定 数..... 93 CONST [例題11] ~ [例題13] 93 [演習問題] (14)~(16) 95
3 章	入力・データの読み込み..... 97 INPUT, INPUT\$, INPUT¥, LINEINPUT, INKEY\$, READ, DATA, RESTORE [例題14] ~ [例題23] 97 [演習問題] (17)~(26) 103
4 章	書式指定の表示と印刷..... 108 PRINT USING, LPRINT USING [例題24] ~ [例題28] 108 [演習問題] (27)~(31) 112
5 章	くり返し..... 115 FOR~NEXT, WHILE~WEND, DO~LOOP WHILE, DO~LOOP UNTIL, DO WHILE ~LOOP, DO UNTIL~LOOP, EXIT FOR, EXIT DO [例題29] ~ [例題38] 115 [演習問題] (32)~(42) 124
6 章	条件分岐..... 128 IF~THEN, IF~THEN~ELSE, ブロックIF, END IF, ELSEIF, OR, AND, XOR, EQV, IMP, NOT [例題39] ~ [例題50] 128 [演習問題] (43)~(56) 138
7 章	多 分 岐..... 145 SELECT~CASE, END SELECT [例題51] ~ [例題55] 145 [演習問題] (57)~(63) 152
8 章	配 列..... 157 DIM, OPTION BASE [例題56] ~ [例題63] 157 [演習問題] (64)~(75) 165

9章	タイプ型..... 172 TYPE, END TYPE [例題64] ~ [例題67] 172 [演習問題] (76)~(80) 177
10章	式の定義..... 181 DEF FN [例題68] ~ [例題72] 181 [演習問題] (81)~(86) 184
11章	プロシジュア・サブルーチン..... 187 DECLARE SUB, SUB, END SUB, CALL [例題73] ~ [例題81] 187 [演習問題] (87)~(97) 196
12章	プロシジュア・ファンクション..... 202 DECLARE FUNCTION, FUNCTION, END FUNCTION [例題82] ~ [例題85] 202 [演習問題] (98)~(103) 206
13章	グローバル変数とローカル変数, プログラム の連結..... 209 COMMON, COMMON SHARED, CHAIN, STATIC, SHARED [例題86] ~ [例題96] 209 [演習問題] (104)~(112) 218
14章	算術関数..... 223 SIN, COS, TAN, ATN, SQR, ABS, LOG, EXP, CINT, CLNG, INT, FIX, SGN, RANDOMIZE, RND, TIMER [例題97] ~ [例題107] 223 [演習問題] (113)~(121) 230
15章	論理演算..... 234 AND, OR, XOR, EQV, IMP, NOT [例題108] ~ [例題113] 234 [演習問題] (122)~(125) 238
16章	文字列関数..... 240 LEFT\$, RIGHT\$, MID\$, LEN, SPACE\$, LTRIM\$, RTRIM\$, MID\$=, INSTR, LCASE\$, UCASE\$, STRING\$ [例題114] ~ [例題122] 240 [演習問題] (126)~(134) 245
17章	漢 字..... 250 LEFT\$, RIGHT\$, KMID\$, KMID\$=, KLEN, KINSTR, KEXT\$, JIS\$, CHR\$, KTN\$ [例題123] ~ [例題134] 250 [演習問題] (137)~(146) 256

18章	桁・行の指定..... 260 LOCATE [例題135] ~ [例題139] 260 [演習問題] (147)~(149) 264
19章	タイマー割り込み・キー割り込み・ サブルーチンへ分岐..... 266 TIMER ON/OFF/STOP, ON TIMER GOSUB~RETURN, KEY()ON/OFF/STOP, ON KEY()GOSUB~RETURN [例題140] ~ [例題144] 266 [演習問題] (150)~(151) 271
20章	シーケンシャルファイル..... 273 OPEN~FOR OUTPUT, OPEN~FOR INPUT, OPEN~FOR APPEND, WRITE#, INPUT#, PRINT#, CLOSE#, EOF [例題145] ~ [例題150] 273 [演習問題] (152)~(160) 279
21章	ランダムファイル(1) 284 OPEN~FOR RANDOM, FIELD, LSET, RSET, MKI\$, MKS\$, MKL\$, MKD\$, CVI, CVS, CVL, CVD, PUT#, GET#, CLOSE# [例題151] ~ [例題158] 284 [演習問題] (161)~(167) 293
22章	ランダムファイル(2) 300 TYPE型を使用 LOF [例題159] ~ [例題165] 300 [演習問題] (168)~(173) 306
23章	シーケンシャルファイルをランダムファイル化... 310 SEEK [例題166] ~ [例題169] 310 [演習問題] (174)~(176) 314
24章	メタコマンド..... 316 REM \$INCLUDE, \$STATIC, \$DYNAMIC [例題170] ~ [例題173] 316 [演習問題] (177)~(181) 320
25章	再帰プログラム..... 324 [例題174] 324 [演習問題] (182)~(184) 326
26章	バージョン4.5で追加された命令..... 328 SOUND, SLEEP, ストップ・キー割り込み, ヘルプ・キー割り込み [例題175] ~ [例題178] 328

Ⅲ グラフィック編

1 章	直線, 四角形, 円, 楕円, 扇形, 点..... 331 SCREEN, CLS, LINE, LINE STEP, LINE~B(F) CIRCLE, PSET, PRESET [例題1] ~ [例題10] 331 [演習問題] (1)~(8) 339
2 章	塗りつぶし; タイルパターン..... 342 PAINT [例題11] ~ [例題12] 342 [演習問題] (9)~(11) 344
3 章	関数式の曲線..... 346 [例題13] ~ [例題14] 346 [演習問題] (12)~(13) 349
4 章	画面切りかえウインドウ・ビュー..... 351 SCREEN, WINDOW, WINDOW SCREEN, VIEW [例題15] ~ [例題18] 351 [演習問題] (14)~(17) 355
5 章	カラー..... 359 PALETTE, COLOR, SCREEN [例題19] ~ [例題20] 359 [演習問題] (18)~(19) 360
6 章	図形の配列へのとりこみと表示..... 362 GET, PUT [例題21] ~ [例題22] 362 [演習問題] (20)~(22) 364
7 章	POINT と DRAW..... 367 POINT, DRAW [例題23] ~ [例題28] 367 [演習問題] (23)~(27) 372

Ⅳ 応用編

1 章	N ₈₈ BASIC (86) との互換..... 375 [応用問題1] ~ [応用問題3] 375
2 章	再帰プログラム..... 378 [応用問題4] ~ [応用問題5] 378

3 章	円弧を使ったグラフ.....	382
	[応用問題 6] ～ [応用問題11]	382
4 章	媒介変数を使った図形.....	400
	[応用問題12] ～ [応用問題20]	400

Quick BASIC, MS-DOS は米国マイクロソフト社の登録商標です。
ATOK 6 は (株) ジャストシステムの登録商標です。

Ver.4.5のメニュー画面

GRPH キーとメニューキー(**F** , **E** , **V** , **S** , **R** , **D** , **C** , **O** , **H**)
をおすことでメニューが選択されます。サブメニューは先頭の1文字をおすか、カーソル
を矢印キーを使って動かし **J** します。

F/ファイル

N/新規
O/読込...
M/結合...
S/保存
A/名前を変えて保存...
V/すべて保存

C/サブファイル作成...
L/サブファイル読込...
U/サブファイル解放...

P/印刷...
D/MS-DOSコマンド

X/終了

E/編集

U/元に戻す
T/カット
C/コピー
P/ペースト
E/削除

S/新規SUB...
F/新規FUNCTION...

GRPH+BS
SHIFT+DEL
CTRL+INS
SHIFT+INS
DEL

V/表示

S/SUB 一覧 F2
E/次のSUB SHIFT+F2
P/画面分割

N/次のステートメント
U/実行画面 F4

I/インクルードファイル編集
L/インクルードファイル表示

S/検索

F/検索...
S/指定文字列検索
R/次を検索
C/置換...
L/ラベル検索...

CTRL+¥
F3

R/実行

S/スタート SHIFT+F5
R/リスタート
N/継続 F5
C/COMMAND\$の編集...

X/実行ファイル作成...
L/ライブラリ作成...

M/メインモジュールの設定...

D/デバッグ

A/ウォッチの追加...
 I/簡易ウォッチ... SHIFT+F9
 W/ウォッチポイント...
 D/ウォッチの削除...
 L/全ウォッチの削除

T/トレース
 H/ヒストリ

B/ブレークポイント F9
 C/全ブレークポイントの削除
 E/エラー時強制中断
 S/次のステートメントの設定

C/関数

O/オプション

D/画面設定...
 S/サーチパス設定...
 R/マウス機能選択...
 *Y/構文チェック
 *F/Fullメニュー

H/ヘルプ

I/索引
 C/目次
 T/キーワード: F1
 H/ヘルプの利用法 SHIFT+F1

(注)

Ver. 4.5ではイージーメニューでたちあがりますが、**GRPH**、**O**でオプションメニューを選び、**F**をおしてフルメニューをセットします。4.2 とほぼ同じメニュー構成です。

(注) Ver. 4.2 と Ver. 4.5 の違い.

Ver.	Ver. 4.5	Ver. 4.2
サブメニュー		
Y/構文チェック	O/オプション	E/編集
O/オプション	O/オプション	V/表示
I/簡易ウォッチ	D/デバッグ	なし
E/エラー時 強制中断	D/デバッグ	なし

言葉の違いは省略

Ver.4.2のメニュー画面

GRPH キーとメニューキー(**F** , **E** , **V** , **S** , **R** , **D** , **C** , **H**)をお
すことでメニューが選択されます。サブメニューは先頭の1文字をおすか、カーソルを矢
印キーを使って動かし **→** します。

F/ファイル

N/新規
O/読込...
M/追加...
S/保存
A/名前を変えて保存...
V/すべて保存

C/サブファイル作成...
L/サブファイル常駐...
U/サブファイル解放...

P/印刷...

D/MS-DOSコマンド

X/終了

E/編集

U/元に戻す
T/カット
C/コピー
P/ペースト
E/削除

S/新規SUB...
F/新規FUNCTION...

Y/構文チェック

GRPH+BS
SHIFT+DEL
CTRL+INS
SHIFT+INS
DEL

V/表示

S/SUB 一覧... f・2
E/次のSUB SHIFT+f・2
P/画面分割

N/次のステートメント
U/実行画面 f・4

I/インクルードファイル編集
L/インクルードファイル表示

O/オプション

S/検索

F/検索...
S/指定文字列検索
R/次を検索
C/置換...
L/ラベル検索...

CTRL+¥
f・3

R/実行

S/スタート SHIFT+f・5
R/リスタート
N/続行 f・5
C/COMMANDS\$ 入力...

X/EXE ファイル作成...
L/ライブラリ作成...

M/メインモジュールの設定...

D/デバッグ

C/関 数

H/ヘルプ(f・1)

A/ウォッチの追加...
W/ウォッチポイント...
D/ウォッチの削除...
L/全ウォッチの削除

T/トレース
H/ヒストリ

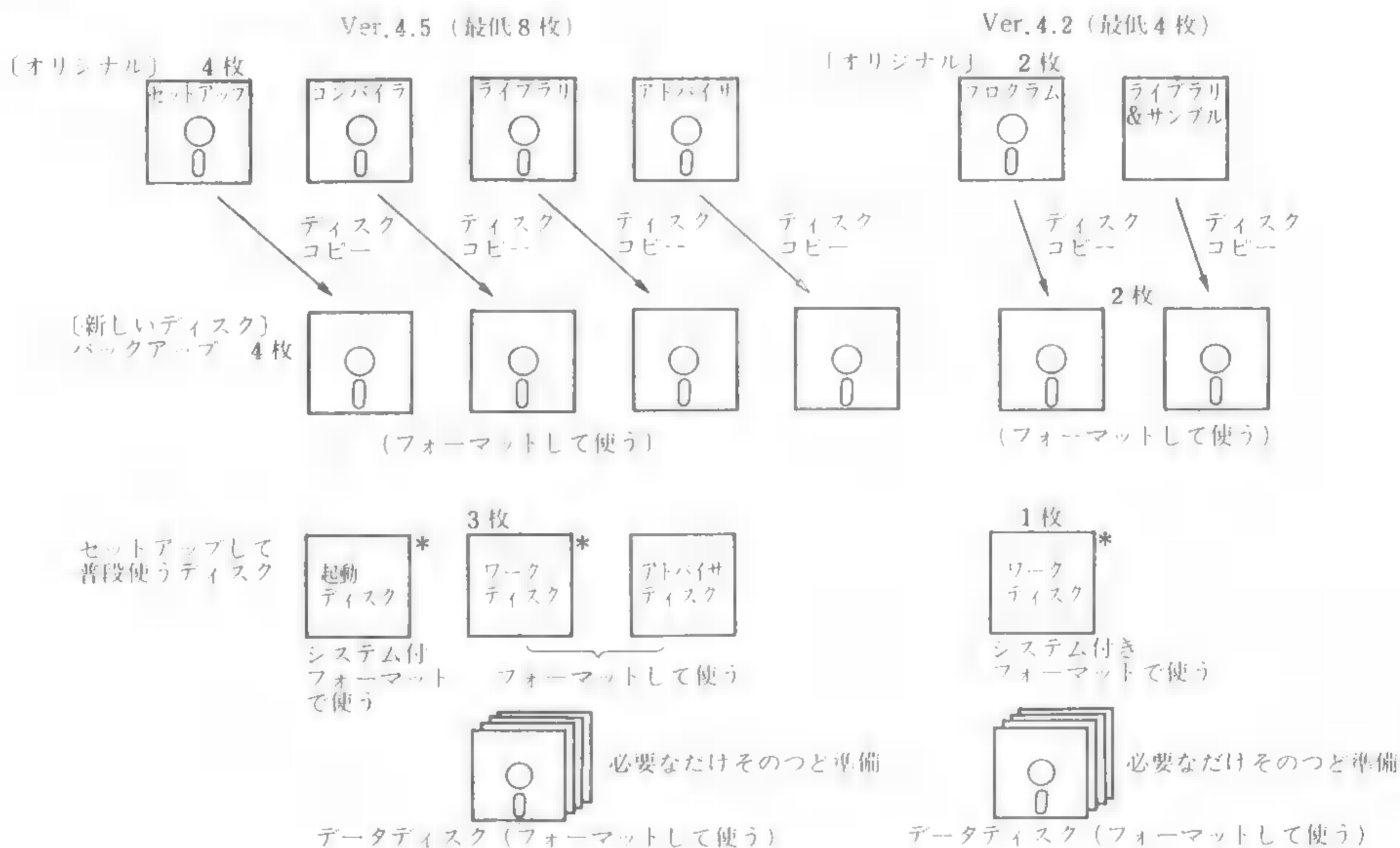
B/ブレークポイント f・9
C/全ブレークポイントの解放
S/カレントステートメントの設定

I.....準備・環境コマンド編



1 章 準 備

▶ セットアップまでに必要なディスク



* Quick BASIC を使うときのメインのディスクを、Ver. 4.5 では『起動ディスク』と呼び、Ver. 4.2 では『ワークディスク』と呼んでいる。Ver. 4.5 では『ワークディスク』の名の別のディスクがあるので、呼び方に注意して下さい。

【例題 1】 バックアップ (メニュー方式によるフォーマットとディスクコピー)

Quick BASIC の 4 枚 (バージョン 4.2 以下では 2 枚) のディスクをバックアップのためコピーせよ。

【解 説】 ① ディスクのバックアップをとります。

① MD-DOS を使用してバックアップをつくります。手順としては、新しいディスクのフォーマット

とコピーです。MS-DOS のバージョンによって若干手続きが異なりますが、本書ではバージョン 3.30 を使って行います。

- ② MS-DOS のディスクを A ドライブに入れ、コンピュータの電源を入れます。次のメニュー画面と異なります。

MS-DOS コマンド メニュー		(コマンド選択) 1/ 6	Menu v2.20
F1	README.DOC ファイルの表示	<div> 日付: 1989-08-20 時刻: 01:28 MS-DOS: Ver. 3.30 </div>	
F2	アプリケーションの登録		
F3	フロッピーディスクの初期化		
F4	フロッピーディスクのバックアップ		
F5	固定ディスクの初期化		
F6	ディスク内全ファイルのコピー		
F7	システムファイルの転送		
F8	ディレクトリ情報の表示		
F9	MENU の終了		
A>TYPE README.DOC			
MS-DOS を使用するにあたっての注意事項を表示します。			
矢印キーで項目を選択し、リターンキーを押してください			

- ③  キーをおして 2 頁目の F8 にカーソルをおきます。

MS-DOS コマンド メニュー		(コマンド選択) 2/ 6	Menu v2.20
F1	フロッピーディスク (640 KBタイプ) の初期化	<div> 日付: 1989-08-20 時刻: 01:34 MS-DOS: Ver. 3.30 </div>	
F2	フロッピーディスク (640 KBタイプ) のバックアップ		
F3	ディスク内全ファイルのコピー (ドライブ指定)		
F4	ファイル内容の表示		
F5	日付の表示と設定		
F6	時刻の表示と設定		
F7	ディスクの初期化 (メニュー形式)		
F8	ディスクのコピー/照合 (メニュー形式)		
F9	フロッピーディスクのバックアップ (ドライブ指定)		
A>FORMAT %: & DISKCOPY %: %:			
フロッピーディスクの複写を作成します。(ドライブの指定ができます)			
矢印キーで項目を選択し、リターンキーを押してください			

- ④  によって次の画面となります。まずフォーマットです。"A:" の部分が反転していますので  で "B:" へ移します。

MS-DOS コマンド メニュー


(ドライブ選択) 1/ 1

Menu v2.20

A:
B:
 C:
 D:
 E:

A>FORMAT B: & DISKCOPY %: %:

矢印キーでドライブを選択し、リターンキーを押してください

- ⑤ により次の画面となります。次はコピーの元のドライブとコピー先のドライブの指定です。まずコピー元の“A:”が反転しています。

MS-DOS コマンド メニュー

(ドライブ選択) 1/ 1

Menu v2.20

A:
 B:
 C:
 D:
 E:

A>FORMAT B: & DISKCOPY A: %:

矢印キーでドライブを選択し、リターンキーを押してください

- ⑥ により次の画面になります。次はコピー先ですから“B:”へ反転を移します。

MS-DOS コマンド メニュー

(ドライブ選択) 1 / 1

Menu v2.20

A:
B:
C:
D:
E:

A>FORMAT B: & DISKCOPY A: B:

矢印キーでドライブを選択し、リターンキーを押してください

- (7) ☐ します。次の画面へ移ります。まずフォーマットをします。

A>FORMAT B:
Format Version 4.10

新しいディスクをドライブ B: に挿入し
どれかキーを押してください

- (8) 上の表示に従い、新しいディスクをドライブ B に入れ ☐ します。次に、ディスクのタイプを問
合わせます。2HD の 1MB タイプを使用するときは ☐ ☐ です。

A>FORMAT B:
Format Version 4.10

新しいディスクをドライブ B: に挿入し
どれかキーを押してください

ディスクのタイプは 1 : 640(KB) 2 : 1(MB) = 2

- (9) フォーマットが始まります。終了すると次の問合わせに入ります。

A>FORMAT B:
Format Version 4.10

新しいディスクをドライブ B: に挿入し
どれかキーを押してください

ディスクのタイプは 1 : 640(KB) 2 : 1(MB) = 2

目的のディスクは 1MB FD です

フォーマットが終了しました
0 10 20 30 40 50 60 70 80 90 100

(%)

1250304 バイト 全ディスク容量
1250304 バイト 使用可能ディスク容量

フォーマット終了時の表示

別のディスクをフォーマットしますか(Y/N) _____ 終了の問合わせ

- ⑩ 今回は4枚フォーマットします。バージョン4.2以下では2枚です。したがって、**[Y]**とします。
- ⑪ Bドライブのディスクをとり、新しいディスクを挿入し **[J]** します。同じようにフォーマットがすむと次のフォーマットの問合わせとなります。

別のディスクをフォーマットしますか(Y/N) **[Y]**

新しいディスクをドライブ B: に挿入し
どれかキーを押してください

目的のディスクは 1MB FD です

フォーマットが終了しました
0 10 20 30 40 50 60 70 80 90 100

(%)

1250304 バイト 全ディスク容量
1250304 バイト 使用可能ディスク容量

別のディスクをフォーマットしますか(Y/N)

- ⑫ **[Y]** をおして、Bドライブから2枚目のディスクをとり3枚目を入れ **[J]** します。フォーマットが終わると次のフォーマットの問合わせとなります。
- ⑬ フォーマットを終了しますので **[N]** とします。

別のディスクをフォーマットしますか(Y/N) **[n]**

- ⑭ バックアップに入ります。バックアップはディスクのコピーです。

別のディスクをフォーマットしますか(Y/N) **[n]**
A> **DISKCOPY A: B:**

DISKCOPY version 4.00

ディスクのコピーを行います

送り側ディスクをドライブ A: に挿入してください
受け側ディスクをドライブ B: に挿入してください
準備ができたならどれかのキーを押してください

- ⑮ AドライブのDOSのディスクをぬき、Quick BASICの“セットアップ”(バージョン4.2以下では“プログラム”)ディスクを入れます。Bドライブでは今フォーマットの終わったディスクが入っていますからそのまま します。
- ⑯ AドライブのディスクがBドライブのディスクにコピーされます。終了すると次の問合わせとなります。 により次のコピーを行います。

A> DISKCOPY A: B:

DISKCOPY version 4.00

ディスクのコピーを行います

コピーは終了しました
もう一度実行しますか (Y/N)Y

- ⑰ ドライブAをQuick BASICの“コンパイラ”(バージョン4.2以下では“ライブラリ”)ディスクに入れかえ、ドライブBは新しいフォーマット済みのディスクに入れかえて します。バージョン4.5では同様に“ライブラリ”ディスク、“アドバイザー”ディスクのコピーをとります。バックアップが終ったらドライブAをDOSのディスクと入れかえ、Bドライブはディスクをとり出します。

DISKCOPY version 4.00

ディスクのコピーを行います

コピーは終了しました
もう一度実行しますか (Y/N)N

コマンド選択に戻ります
準備ができたなら、どれかキーを押してください

- ⑱ キーをおすとDOSのメニュー画面へもどります。

【例題2】 ファイルの確認

バックアップされたディスクのファイルを確認せよ。

【解 説】 ㊦ バックアップファイルの内容の確認をします。

- ① バックアップされたディスクのファイルを確認します。次のようになっていればコピーは完全です。

[Ver.4.5]

ドライブ B: のディスクのボリュームラベルはありません。
ディレクトリは B:¥

(セットアップ・
ディスク)

PACKING	LST	13570	89-10-17	18:09
QBE	DOC	4475	89-09-06	18:33
README	DOC	4673	89-10-13	15:40
SAMPLE	DOC	2942	89-10-13	14:22
SETUP	EXE	61441	89-10-13	19:12
SETUP	@@@	3244	89-10-13	19:53
SAMPLE	<DIR>		89-10-18	16:27
LEARN	<DIR>		89-10-18	16:29
TRD07	TXT	8425	89-10-17	22:44

9 個のファイルがあります。
690176 バイトが使用可能です。

ドライブ B: のディスクのボリュームラベルはありません。
ディレクトリは B:¥

(コンパイラ・
ディスク)

BIN <DIR> 89-10-18 16:29
MOUSE <DIR> 89-10-18 16:30

2 個のファイルがあります。
107520 バイトが使用可能です。

ドライブ B: のディスクのボリュームラベルはありません。
ディレクトリは B:¥

(ライブラリ・
ディスク)

LIB <DIR> 89-10-18 16:32
USERLIB <DIR> 89-10-18 16:33

2 個のファイルがあります。
545792 バイトが使用可能です。

ドライブ B: のディスクのボリュームラベルはありません。
ディレクトリは B:¥

(アドバイザ・
ディスク)

QBADVR <DIR> 89-10-18 16:35

1 個のファイルがあります。
591872 バイトが使用可能です。

Ver.4.2]

QB_ENV <DIR> 88-12-15 4:20
QB_CMP <DIR> 88-12-15 4:20
LIB_E <DIR> 88-12-15 4:20
SETUP DOC 2763 88-12-15 4:20
SETUP EXE 30198 88-12-15 4:20
PACKING LST 3303 89-01-15 4:20
README DOC 7832 89-01-15 4:20
SAMPLE DOC 4196 89-01-15 4:20

(プログラムディスク)

8 個のファイルがあります。
176128 バイトが使用可能です。

LIB_A <DIR> 88-12-15 4:20
SOURCE <DIR> 88-12-15 4:20

(ライブラリディスク)

2 個のファイルがあります。
818176 バイトが使用可能です。

- ② 確認は次のようにします。MS-DOS 3.30 の場合はメニュー画面から“F8”を選びます。

MS-DOS コマンド メニュー	(コマンド選択) 1/ 6	Menu v2.20
<p>F1 README.DOCファイルの表示</p> <p>F2 アプリケーションの登録</p> <p>F3 フロッピーディスクの初期化</p> <p>F4 フロッピーディスクのバックアップ</p> <p>F5 固定ディスクの初期化</p> <p>F6 ディスク内全ファイルのコピー</p> <p>F7 システムファイルの転送</p> <p>F8 ディレクトリ情報の表示</p> <p>F9 MENUの終了</p>	<div style="border: 1px solid black; padding: 5px;"> <p>日付: 1989-08-20</p> <p>時刻: 03:20</p> <p>MS-DOS: Ver. 3.30</p> </div>	

A>DIR %: /P

ディレクトリの内容を表示します。

矢印キーで項目を選択し、リターンキーを押してください

- ③ ☐ をします。次の画面で“B:”を選び、ディスクをBドライブに入れ ☐ します。“A:”を選んでディスクをAドライブに入れて ☐ しても同じです。

MS-DOS コマンド メニュー	(ドライブ選択) 1/ 1	Menu v2.20
<p>A:</p> <p>B:</p> <p>C:</p> <p>D:</p> <p>E:</p>		

A>DIR B: /P

矢印キーでドライブを選択し、リターンキーを押してください

- ④ メニューを使わない場合はBドライブにディスクを入れて

DIR B: ☐

あるいはAドライブにディスクを入れて

DIR A: ☐ です。

【例題3】 フォーマット（ふだん使う Quick BASIC 用ディスク）

新しいディスクをフォーマットせよ。

【解 説】 ① フォーマットをします。

① 新しいディスクのフォーマットをします。

② A ドライブに MS-DOS のディスクを入れ電源を入れます。メニュー方式によるフォーマット例は【例題1】のとおりです。ここではメニュー方式によらない例を示します。メニューから **[F9]** をおします。

A>

の表示となります。

③ 次のようにかき **[J]** します。

A>FORMAT B: **[J]**

④ 【例題1】の項⑦となりますので、B ドライブに新しいディスクを入れ **[J]** します。続いて 1MB の指定のため **[2]** **[J]** とします。

⑤ フォーマットが終ると次のフォーマットをするかどうかを問合わせます。

別のディスクをフォーマットしますか(Y/N) y

⑥ **[Y]** をおすと同じ手順で、新しいディスクを B ドライブに入れ **[J]** , **[2]** **[J]** とします。

⑦ 3 枚フォーマットをして下さい。バージョン 4.5 では 2 枚は Quick BASIC のワークディスクとアドバイザディスク用、1 枚は Quick BASIC のデータ用です。バージョン 4.2 以下ではデータ用のみ 1 枚です。フォーマットを終了したとき、別のディスクをフォーマットしますか[Y/N]で **[N]** をおすと終了します。

【例題4】 バックアップ（非メニュー方式）

Quick BASIC の 4 枚（バージョン 4.2 以下では 2 枚）のディスクをバックアップコピーせよ。

【解 説】 ① バックアップコピーをとります。

① メニュー方式を使わずに、バックアップ・コピーをとります。

② メニューから F9 をおし A>とします。A>から次のようにかきます。

A>DISKCOPY A:B:

A ドライブから B ドライブへコピーする命令です。 **[J]** します。

③ 次の表示になりますので A ドライブに Quick BASIC の”セットアップ”（バージョン 4.2 以下では”プログラム”）ディスクを入れ、B ドライブにフォーマットしたディスクを入れ **[J]** します。

DISKCOPY version 4.00

ディスクのコピーを行います

送り側ディスクをドライブ A: に挿入してください
受け側ディスクをドライブ B: に挿入してください
準備ができたらかのキーを押してください

- ④ ディスクのコピーが終ると次の表示となり次のコピーをするかどうか問合わせます。

コピーは終了しました
もう一度実行しますか (Y/N) Y

- ⑤ ☐ Y をおすと項③へもどりますので A ドライブに Quick BASIC の"コンパイラ" (バージョン 4.2 以下では"ライブラリ") ディスク, B ドライブにフォーマットした別の新しいディスクを入れて ☐ します。

コピーが終ると項④の問合わせとなります。バージョン 4.5 では同様に"ライブラリ"ディスク, "アドバイザー"ディスクをコピーします。コピーが終ったら次へすすみます。

- ⑥ A ドライブに DOS のディスクを入れ, B ドライブはコピー済みのディスクを抜いて ☐ N とします。バックアップは終了です。

【例題 5】 システム付フォーマット

新しいディスクをシステム付きでフォーマットせよ。

【解 説】 ① システム付フォーマットをします。

- ① Quick BASIC の起動ディスク (バージョン 4.2 以下ではワークディスク) をつくるディスクはシステム付きのディスクにセットアップする必要があります。そのためシステム付フォーマットを 1 枚行います。

- ② DOS のディスクを A ドライブに入れ電源を入れます。メニュー画面で ☐ f9 をおし A>とし, A>で次のように記入します。

A>FORMAT B:/S

- ③ ☐ によって, 【例題 1】と同じ手順でフォーマットが行われ, 終了します。

Format Version 4.10

新しいディスクをドライブ B: に挿入し ————— 新しいディスクを B
どれかキーを押してください ドライブに入れ ☐

ディスクのタイプは 1 : 640(KB) 2 : 1(MB) = ☐ 2 ☐

目的のディスクは 1MB FD です

フォーマットが終了しました

0 10 20 30 40 50 60 70 80 90 100
(%)

システムを転送しました

```
1250304 バイト 全ディスク容量
120832 バイト システム領域
1129472 バイト 使用可能ディスク容量
```

別のディスクをフォーマットしますか(Y/N)

☐ N で終了

- ④ このディスクを DIR でみると COMMAND.COM というファイルが 1 つ入っています（正確には他に 2 つのかくれたファイルがコピーされています）。

A>dir b:

ドライブ B: のディスクのボリュームラベルはありません。
ディレクトリは B:¥

```
COMMAND  COM      24931  88-07-13      0:00
      1 個のファイルがあります。
1129472 バイトが使用可能です。
```

これをシステム付きディスクといいます。このディスクを使うと DOS を立ちあがらすことができます。【例題 1】でつくったディスクはシステムがありません。したがって、DIR でみるとファイルが入っていません。

A>dir b:

ドライブ B: のディスクのボリュームラベルはありません。
ディレクトリは B:¥

ファイルが見つかりません。

【例題 6】 セットアップ

Quick BASIC をセットアップせよ。

【解 説】 ◎ Quick BASIC をセットアップします。

- ① DOS のディスクを A ドライブに入れ、電源を入れて、DOS を立ちあがらせます。日付け、11時をセット（または 2 回 キーをおす）すると次の A プロンプトとなります（メニュー方式の場合は をおす）。

A>

- ② まずバージョン 4.5 でのセットアップを説明します。新しいシステム付きフォーマット済みのディスク 1 枚と、システムなしディスク 2 枚を用意します。
- ③ A ドライブにセットアップディスクを入れます。次のように入力します。

A>SETUP

- ④ あとは画面の指示に従うだけですが、順を追って見ていきます。
まずセットアップユーティリティによりメニューが出ます。

Quick BASIC セットアップ ユーティリティ

Version 1.00

***** < メニュー > *****

1. Quick BASICの利用環境を作成します。
2. QBの概要、操作方法の説明を行います。
3. セットアップを終了します。

●メニューを選択して下さい。・・・・・・？ 1

Quick BASICの利用環境を作成します。
 始めてお使いになる方は必ず 1のメニューを選んでください。
 途中でセットアップを中断するには、CTRL+CもしくはSTOPキーを押して
 ください。この場合は、再度セットアップを起動して最初から行う必要があります。

2のメニューを選択すると、Quick BASICの利用環境の作成は行わずに、
 QBの概要や操作方法を説明する「QBチュートリアル」が起動します。
 利用環境の設定を行う場合は1を選択してください。

1の「Quick BASICの利用環境を作成します」を選ぶため、「1」を入力します。 [1] [↵] です。

⑤

Quick BASIC セットアップ ユーティリティ

Version 1.00

●転送元のドライブ名を設定してください？ A

Quick BASICのマスターディスクの入っているドライブ名を設定してください。

オリジナルディスクの入るドライブを指定します。 [A] [↵] とします。オリジナルディスクは常に
 Aドライブに入れ、ふだん使うディスクはBドライブに入れてセットアップするものとします。

⑥

Quick BASIC セットアップ ユーティリティ

Version 1.00

- 転送元のドライブ名を設定してください？ A
- どのドライブにインストールしますか？ B

Quick BASIC をインストールするドライブ名を入力してください。
 フロッピーディスクにインストールする場合は、以下の準備が整っているしている
 ことを確認してください。

1. フォーマット済みフロッピーディスクの準備。
 (DD 版の時は 4 枚、HD の時は 3 枚のディスクが必要です。)
 2. パッケージに添付されている黒の作業ディスクラベルを貼る。
 3. [QB 起動ディスク] に MS-DOS を転送する。
- 準備ができていない場合は、CTRL+C でセットアップを終了し、ディスクを用意してから再度セットアップを起動してください。

セットアップするドライブを指定します。B ドライブとします。[B] [↓] です。

⑦

Quick BASIC セットアップ ユーティリティ

Version 1.00

- 転送元のドライブ名を設定してください？ A
- どのドライブにインストールしますか？ B

- 上記の設定でよろしいですか[Y/N]？ Y

上記の設定でよければ Y を、変更したければ N を指定してください。
 N が指定された場合は、この画面の最初から指定しなおすことができます。
 この時、前回の設定がデフォルト値として表示されます。

設定の可否を確認します。A ドライブをオリジナル、B ドライブにセットアップするディスクを入
 れますので、[Y] [↓] とします。

⑧

QuickBASIC セットアップ ユーティリティ

Version 1.00

■ 数値演算ライブラリを選択してください [1:Alt Math / 2:Emulator Math]? 1

QuickBASICは、2種類の数値演算ライブラリをサポートしています。
 ここでは、数値演算コプロセッサを装着した機種上で非常に高速な演算処理を保証
 するエミュレータライブラリ(Emulator Math)と、装着/非装着にかかわらずある程
 度のスピードで動作する代替演算ライブラリ(Alt Math)のどちらを使うかを選択し
 ます。

Emulator 版の QuickBASIC インタープリタのファイル名は QBE.EXE です。プログ
 ラム名が異なることにご注意ください。なお、セットアップディスクの QBE.DOC に
 は、より詳しい情報が記されています。Emulator 版をご使用の場合には必ずお読
 みいただくようお願いいたします。

数値演算ライブラリを問合わせます。通常は です。

⑨

QuickBASIC セットアップ ユーティリティ

Version 1.00

- 数値演算ライブラリを選択してください [1:Alt Math / 2:Emulator Math]? 1
- マウスを使用しますか [Y/N]? Y

マウスを利用することによって、統合環境の操作性がさらに向上します。ご利用を
 お勧めします。
 なお、マウスを利用する場合は、必ずマイクロソフトのマウスイバ Version 6
 を利用してください。他のバージョンのマウスイバでの利用は QuickBASIC
 の動作を保証できません。

マウスの使用または不使用を入力します。使用するときは です。

(10)

Quick BASIC セットアップ ユーティリティ

Version 1.00

- 数値演算ライブラリを選択してください [1:Alt Math / 2:Emulator Math]? 1
- マウスを使用しますか [Y/N]? Y
- マウスの種類を選択してください [1:バスマウス / 2:シリアルマウス] 1

ご使用のマウスの種類によって、1 もしくは 2 を選択してください。
 RS-232C ポートに接続されているマウスは 2:シリアルマウス、それ以外のものは、1:バスマウスを選択してください。

マウスを使用するときマウスの種類を問合わせます。バスマウスのときは **1** **↓** , シリアルマウスのときは **2** **↓** です。

(11)

Quick BASIC セットアップ ユーティリティ

Version 1.00

- 数値演算ライブラリを選択してください [1:Alt Math / 2:Emulator Math]? 1
- マウスを使用しますか [Y/N]? Y
- マウスの種類を選択してください [1:バスマウス / 2:シリアルマウス] 1
- マウスドライバを選択して下さい [1:MOUSE.SYS 2:MOUSE.COM]? 2

マウスドライバには、CONFIG.SYS でデバイスドライバとしてメモリに組み込まれる 1:MOUSE.SYS と必要に応じて常駐、解放可能な 2:MOUSE.COM があります。どちらを利用していただくかは、メモリ解放ができるため MOUSE.COM を選択されることをお勧めします。
 マルチプラン 3.1 をご使用の方は、1:MOUSE.SYS を必ず選択してください。

マウスドライバを入力します。通常は **2** **↓** です。

⑫

Quick BASIC セットアップ ユーティリティ

Version 1.00

- 数値演算ライブラリを選択してください[1:Alt Math / 2:Emulator Math]? 1
- マウスを使用しますか[Y/N]? Y
- マウスの種類を選択してください[1:バスマウス / 2:シリアルマウス] 1
- マウスドライバを選択して下さい[1:MOUSE.SYS 2:MOUSE.COM]? 2
- ドキュメントファイルを転送しますか[Y/N]? Y

Quick BASICには、README.DOC、SAMPLE.DOC、QBE.DOCなど Quick BASICをお使いになるうえで、重要なドキュメントファイルが含まれています。Yを選択した場合は、QBワークディスクもしくはハードディスクのDOCディレクトリに転送されます。必ずお読みいただくようお願いします。

ドキュメントファイルのコピーをするかどうかを問合わせます。ドキュメントファイルはQuick BASICを使う上での注意事項等がかいてあります。コピーをする必要はありませんが、一読する必要はあります。既定値は Y ↓ です。

⑬

Quick BASIC セットアップ ユーティリティ

Version 1.00

- 数値演算ライブラリを選択してください[1:Alt Math / 2:Emulator Math]? 1
- マウスを使用しますか[Y/N]? Y
- マウスの種類を選択してください[1:バスマウス / 2:シリアルマウス] 1
- マウスドライバを選択して下さい[1:MOUSE.SYS 2:MOUSE.COM]? 2
- ドキュメントファイルを転送しますか[Y/N]? Y
- サンプルプログラムを転送しますか[Y/N]? Y

Quick BASICのパッケージには、多数のサンプルプログラムや便利なライブラリが含まれています。プログラミングの参考例やツールとしてご利用頂ければ幸いです。なお、サンプルプログラムの説明は、ドキュメントファイル SAMPLE.DOCに記述されています。

サンプルプログラムを転送するかどうかを問合わせます。サンプルプログラムの転送は必ずしも必要はありませんが、転送するとコマンド類の説明がわかりやすくなったり、サンプルプログラムを利

用したりすることができます。既定値は ☐ Y ☐ N です。

⑭

Q u i c k B A S I C セットアップ ユーティリティ

Version 1.00

- 数値演算ライブラリを選択してください[1:Alt Math / 2:Emulator Math]? 1
- マウスを使用しますか[Y/N]? Y
- マウスの種類を ☐ 択してください[1:バスマウス / 2:シリアルマウス] 1
- マウスドライバを選択して下さい[1:MOUSE.SYS 2:MOUSE.COM]? 2
- ドキュメントファイルを転送しますか[Y/N]? Y
- サンプルプログラムを転送しますか[Y/N]? Y

● 上記の設定でよろしいですか[Y/N]? Y

 上記の設定でよければ Y を、変更したければ N を指定してください。
 N が指定された場合は、この画面の最初から指定しなおすことができます。
 この時、前回の設定がデフォルト値として表示されます。

以上の設定の確認を行います。良い場合は ☐ Y ☐ N です。悪い場合は ☐ N ☐ Y です。☐ N の場合は最初から入力しなおします。

⑮ 続いて、次の表示に従って、“プログラムディスク”を A ドライブに、“起動ディスク”を B ドライブに入れ ☐ N ☐ Y します。¥BIN¥QB.EXE 以下のファイルを B ドライブへコピーします。

Q u i c k B A S I C セットアップ ユーティリティ

Version 1.00

- 『マスターディスク』
- ドライブ『 A 』に『 プログラムディスク 』をセットして下さい。
- 『ユーザーディスク』
- ドライブ『 B 』に『 QB 起動ディスク 』をセットして下さい。
- 【何かキーを押して下さい】

⑯ 以降は画面の指示でディスクを入れかえます。セットアップの手続きがおわると ④ へもどります

ので、**③** **␣** でセットアップは終了です。

起動ディスクが Quick BASIC を使うときのメインのディスクです。

- ⑰ バージョン 4.2 でのセットアップは次の手順となります。Quick BASIC のプログラムディスクを B ドライブに入れます。B:setup **␣** とします。まず Quick BASIC セットアップユーティリティの説明が表示されます。続いてセットアップに関する質問が続きます。

セットアップを開始しますか[Y/N]? **Y** **␣**

ハードディスクにセットアップしますか[Y/N]? **N** **␣**

Quick BASIC をどのドライブにインストールしますか? **A** **␣**

マウスを使用しますか[Y/N]? **Y** **␣**

マウスの種類を選択して下さい[1:パス/2:シリアル]? **1** **␣**

マウスドライバを選択して下さい[1:MOUSE.SYS/2:MOUSE.COM]? **2** **␣**

上記の設定でよろしいですか[Y/N]? **Y** **␣**

以上によってセットアップが始まります。今回の例は上のようにしていますが自分の組合わせ設定を行います。この中の“Quick BASIC をどのドライブにインストールしますか”はふだん使うワークディスクの入っているドライブを指定します。今回はワークディスクを A ドライブに入れます。

後は画面の指示どおり行って下さい。

セットアップが終了します。

ワークディスクのみ A ドライブにのこします。今後はこれが Quick BASIC を使うときのメインのディスクとなります。

- ⑱ なお、データを収納するディスクは、バージョン 4.5 でも 4.2 でもフォーマット（システムなし）済みのディスクです。

【例題 7】 プリンタのインストール

プリンタのインストールをせよ。

【解 説】 ㉟ プリンタのインストールをします。

- ① DOS のバージョンによっては印刷ができないものがあります。次の表のとおりです。たとえば、DOS バージョン 3.3 ではこのままでは印刷ができません。次に印刷のできるようにします。

MS-DOS のバージョン	PRINT.SYS
MS-DOS Ver.2.0 (PS98-121-XXX)	不要
(PS98-122-XXX)	〃
(PS98-123-XXX)	〃
MS-DOS Ver.3.1 (PS98-125-XXX)	〃
(PS98-127-XXX)	〃
(PS98-129-XXX)	〃
(PS98-011-XXX)	必要
MS-DOS Ver.3.3 (PS98-013-XXX)	〃

- ② MS-DOS の PRINT.SYS と PRINT.EXE のファイルをコピーします。DOS を A ドライブに入れます。まず DIR A: **␣** によって PRINT.SYS および PRINT.EXE のファイルがあることを確認

します。別々のディスクに入っているものがあります。その場合は DOS のディスクをそのつど交換してください。まず、PRINT.EXE があつたとします。B ドライブにはバージョン 4.5 では起動ディスク 4.2 ではワークディスクを入れます。

COPY A:PRINT.EXE B: です。

次に A ドライブに PRINT.SYS の入っているディスクを入れます。

COPY A:PRINT.SYS B: です。

これによって 2 個のファイルが起動ディスクまたはワークディスクへコピーされました。

③ 次に CONFIG.SYS の内容を変更します。

A プロンプトから次の手順で CONFIG.SYS の内容をかきかえて保存しなおします。

A>QB

クイックベーシックの初期画面となります。

, , ,

次の CONFIG.SYS の内容が表示されます。

Ver.4.5

```
SHELL=A:¥COMMAND.COM A:¥ /P
FILES=20
BUFFERS = 10
```

Ver.4.2

```
FILES=20
BUFFERS = 20
```

カーソルを最後の行へおき次のようにかき加えます。

Ver.4.5

```
SHELL=A:¥COMMAND.COM A:¥ /P
FILES=20
BUFFERS = 10
DEVICE = PRINT.SYS
```

Ver.4.2

```
FILES=20
BUFFERS = 20
device = print.sys
```

, , によって、新しい CONFIG.SYS の内容が保存されました。これでプリンタが使えます。一度電源をおとします。

【例題 8】 日本語フロントエンドプロセッサの組み込み

日本語フロントエンドプロセッサを組み込み。

【解 説】 ◎ 日本語フロントエンドプロセッサを組み込みます。

① 日本語を使うためには日本語フロントエンドプロセッサを組み込む必要があります。この手順は

【例題 7】で印刷のために関連のファイルを組み込み、CONFIG の内容をかきかえたのと同様です。

② ここでは ATOK を組み込みます。ATOK の入っている一太郎等のディスクを準備してください。

DOS を A ドライブに入れ電源を入れて、日、時を入力して DOS をたちあげ A プロンプトとします。

A ドライブに ATOK6A.SYS, ATOK6B.SYS の 2 つのファイルのあるディスクを入れます。B ドライブにワークディスク（バージョン 4.5 でも 4.2 でもワークディスク）を入れます。

COPY A:ATOK6A.SYS B:

COPY A:ATOK6B.SYS B:

この2つによって2つのファイルがBドライブのディスクにかきこまれます。

- ③ なお、次のように1度でコピーすることもできます。

```
A>copy a:atok6*.* b:
A:ATOK6A.SYS
A:ATOK6B.SYS
```

2 個のファイルをコピーしました。

- ④ 次に ATOK.DIC ですが、これは漢字の辞書で非常に量が多いので起動またはワークディスクに入りません。したがって、データディスクにコピーします。

A ドライブに ATOK.DIC の入っているディスク、B ドライブにデータディスクを入れます。

COPY ATOK.DIC B: [J] によりコピーされます。

```
A>copy a:atok.dic b:
```

1 個のファイルをコピーしました。

- ⑤ 次に CONFIG.SYS をかきなおします。まずバージョン 4.5 では ATOK をワークディスクにかきましたので次のようにします。起動ディスク（バージョン 4.5）を A ドライブに入れ、電源を一度きって入れなおして DOS を立ちあげます。A プロンプトから次のようにします。

```
A>[Q][B][J]
```

Quick BASIC がたちあがります。Quick BASIC の初期画面となります。

[GRPH], [F], [O], [C][O][N][F][I][G][.][S][Y][S][J] により CONFIG.SYS の内容を表示させます。次のようにかき加えます。

```
DEVICE = B:ATOK6A.SYS / D = B / S = 1 / E = 1 / B = 0
DEVICE = B: ATOK6B.SYS
```

[GRPH], [F], [S] でこの内容が保存されます。これで起動ディスク（バージョン 4.5）は完成です。

- ⑥ バージョン 4.2 ではワークディスクに ATOK をかきましたので、次のようにします。ワークディスク（バージョン 4.2）を A ドライブに入れ、項⑤と同様にして Quick BASIC を立ち上げらせ、CONFIG.SYS を次のようにかきかえ保存します。

```
device = atok6a.sys / d = b / s = 1 / e = 1 / b = 0
device = atok6b.sys
```

- ⑦ なおこのディスクをそのままコピーしておくと、今後ディスクのセットアップをしなくて済みます。

【例題 9】 Quick BASIC のエディタモード

Quick BASIC の初期画面エディタモードをつくれ。

【解 説】 ◎ Quick BASIC 初期画面エディタモードをつくります。

- ① 起動ディスク（バージョン 4.5）またはワークディスク（バージョン 4.2）を A ドライブに入れ電源を入れます。バージョン 4.5 を入れた場合は B ドライブにワークディスク（バージョン 4.5）を入れます。プリンタの使用、日本語 ATOK6 の使用、マウスの使用が可能となります。最後は A プロンプトです。

② **Q** **B** **J** とします。バージョン 4.5 では次の表示となり **ESC** をおすと初期画面となります。

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ O/オプション H/ヘルプ
Untitled

Welcome to

Microsoft (R) QuickBASIC Version 4.50J

(C) Copyright Microsoft Corporation, 1985-1989.
All rights reserved.

現在, Easyメニューで起動しています

< RETURNキーで QB アドバイザが表示されます >

< **ESCキーで編集画面に移ります** >

ダイレクト モード

<SHIFT+F1=ヘルプ> <F6=窓切換> <F2=SUB一覧> <F5=実行> <F8=トレース> ! 00001:001

バージョン 4.2 ではすぐ初期画面となります。これはプログラムの作成を行うエディットモードです。

Ver.4.2

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 H/ヘルプ (f.1)
<Untitled>

□

□ ::
ダイレクト モード

メインプログラム: <Untitled> 動作状況: 実行されていません 00001:001

③ **CTRL** + **XFER** キーをおすと次のようになります。日本語が使えることがわかります。日本語を使うときは ATOK.DIC の入ったディスクを B ドライブに入れます。

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 H/ヘルプ (f.1)
<Untitled>

□ ::
ダイレクト モード

読み込み中

2章 プログラムの作成・ランタイム実行・保存・終了

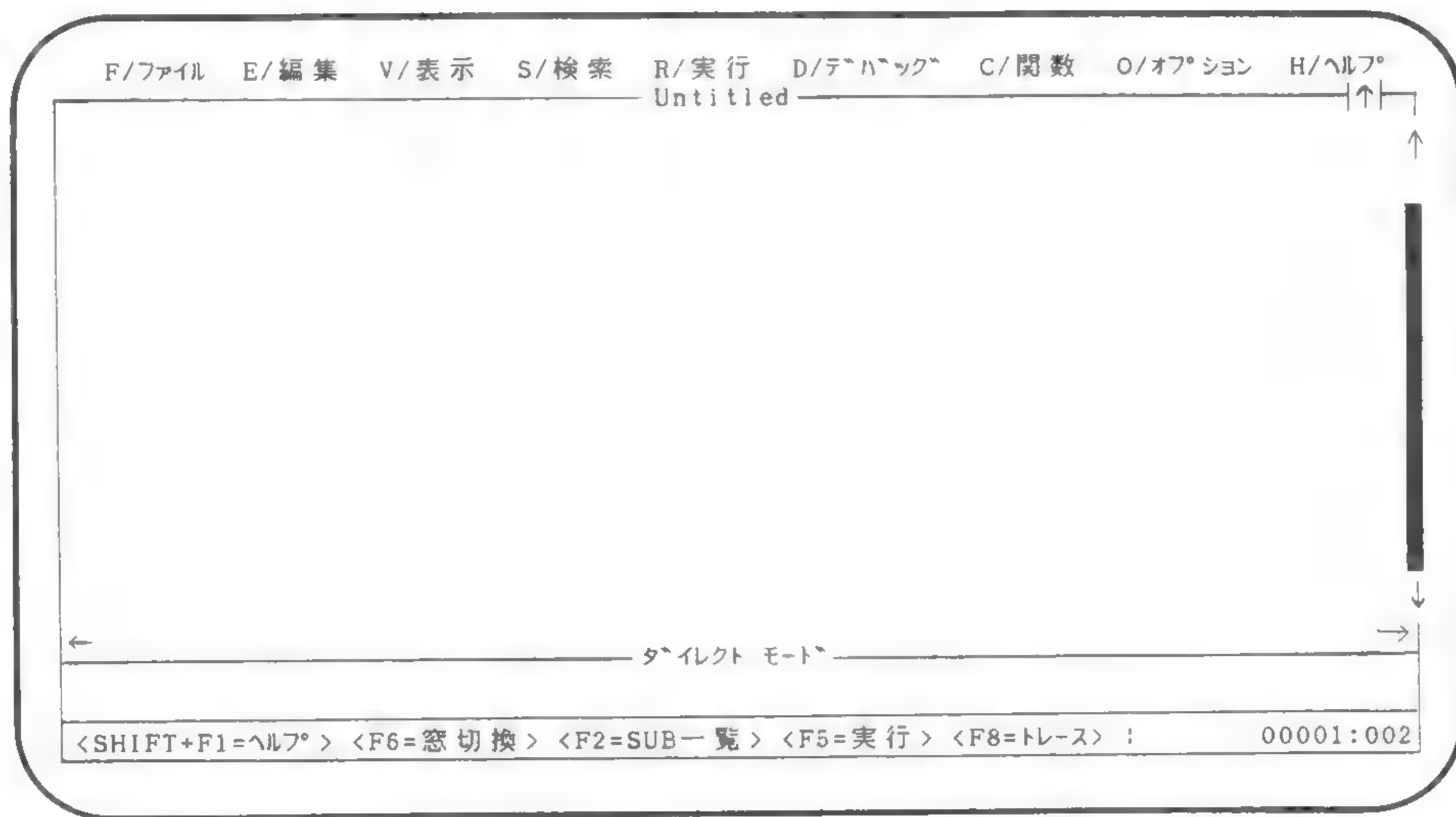
【例題 10】 テキスト画面にプログラムをかく

数 x , y を入力して、積を z に代入し、 x の値 $\times y$ の値 $= z$ を表示するプログラムをつくれ。

【解 説】 ㊦ テキスト画面にプログラムをかきます。

- ① Quick BASIC をはしらせた画面は編集画面で、テキスト画面です。ここにプログラムを記入します。小文字で入力をします。何かすでに記入されているような場合はファイルメニューの“N/新規”を選びます。

Ver.4.5



- ② まず、 x と y の入力ですから次のように入力します。行番号は必要ありません。これが従来の BASIC と大きく異なります。

```
input x,y
```

- ③ ☐ とします。input が大文字となり、変数 x , y は小文字のままです。次のようになります。

```
INPUT x, y
```

- ④ 次に積を z に求めます。 $z = x * y$ です。次のようにかきます。

```
INPUT x, y  
z=x*y
```

- ⑤  をします。変数は小文字ですからそのままですが、見やすくスペースがとられます。

```
INPUT x, y
z = x * y
```

- ⑥ 次に表示の命令をかきます。次のようになります。

```
INPUT x, y
z = x * y
print x; "*"; y; "="; z
```

- ⑦  をおします。次のように **print** 命令は大文字となります。

```
INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
```

- ⑧ これで終了ですから **end** をかきます。

```
INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
end
```

- ⑨  をおします。次のようになります。これでプログラムができました。

```
INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
END
```

【例題 11】 プログラムの保存

【例題 10】でつくったプログラムをプログラム名 **"renshu-1"** でディスクに保存せよ。

【解 説】 ㉟ ファイル（プログラム）を保存します。

- ① 【例題 10】でつくったプログラムをディスクに保存します。プログラムに誤りがあるかも知れないときでも、プログラムを実行して、重大な誤りのためコンピュータが暴走し、復帰できなくなることがあります。このときリセットするとメモリのプログラムは消去されますから、一応保存しておくことをおすすめします。完成したとき、再び同じ名前で保存しなおします。
B ドライブにはデータディスクを入れておきます。
- ② ファイルメニューを選択します。プログラムは Quick BASIC ではファイルとして扱いますので、今後はプログラム名ではなくファイル名と呼びます。ファイルはプログラムだけでなくデータの入っているファイルもありますから、ここではプログラムの入っているファイルと理解します。
- ③ "A/名前を変えて保存..." を選びます。次の画面となります。すなわち、ファイル名の入力、ファイルの保存形式の入力です。

Ver.4.5

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 O/オプション H/ヘルプ

Untitled

A/名前を変えて保存

ファイル名を設定してください

N/ファイル名: _____ ファイル名をかく。BドライブのときはB:ファイル名とする。

A:¥

D/ディレクトリ

BIN
LIB
[-A-]
[-B-]

ファイル形式

(*) Q/QuickBASIC
標準ファイル

() T/テキストファイル

[TAB] キーで項目間をカーソル
が動く。同じ項目の中は矢印キ
ーで選択

< 確認 > < 取消 > < H/ヘルプ >

F1=ヘルプ RETURN=実行 ESC=取消 TAB=次項目 カーソルキー=選択 00001:001

④ まずファイル名を入力します。カーソルはN/ファイル名:の後にあります。

今回はrenshu-1ですからb) [: r e n s h u - 1]とします。[↓]をおします。b:はB
ドライブのディスクに格納することを示します。省略するとカレントドライブへ入ります。

Ver.4.5

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 O/オプション H/ヘルプ

Untitled

A/名前を変えて保存

ファイル名を設定してください

N/ファイル名: **b:renshu-1**

A:¥

D/ディレクトリ

BIN
LIB
[-A-]
[-B-]

ファイル形式

(*) Q/QuickBASIC
標準ファイル

() T/テキストファイル

< 確認 > < 取消 > < H/ヘルプ >

F1=ヘルプ RETURN=実行 ESC=取消 TAB=次項目 カーソルキー=選択 00001:001

⑤ 次に[TAB]キーをおして(*)Q/標準ファイル"*"にカーソルを移します。[↓]キーをおすと下の
()T/テキストファイルの()の中に*がつきます。再度[↓]キーをおすともとへもどります。どち
らか決めて[↓]します。

[Ver.4.5]

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 O/オプション H/ヘルプ

RENSH-1.BAS

```

INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
END

```

Q/標準ファイルは名前のおり標準ファイルです。T/テキストファイルはテキストファイルです。今回はどちらで格納してもかまいません。詳しくはMS-DOSの説明書をみてください。

renshu-1 のファイルが B ドライブのディスクにできます。このとき、拡張子 .BAS がつきます。拡張子 BAS は BASIC のプログラムを意味します。ファイル名が画面中央上部に表示されます。大文字で表示されますが、大文字、小文字の区別はありません。

【例題 12】 プログラムのランタイム実行

【例題 10】のプログラムをランタイムで実行せよ。

```

INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
END

```

【解 説】 ① プログラムをランタイムで実行

- ① プログラムの実行はランタイムの実行と EXE ファイルを作成して DOS ベースで実行する 2 つの方法があります。プログラムができたならランタイムで実行して、解答をみてプログラムのでき具合をチェックします。この結果によって、EXE ファイルをつくるというのが普通の手順です。
- ② ランタイムの実行は、プログラムを簡単に実行する方法です。
- ③ プログラムが画面に表示されているものとします。

f・5 キーをおします。プログラムが実行されます。この場合は、数 x と y の入力ですから?が表示されます。値を 2 つ入力します。値の間はカンマで区切ります。たとえば 3, 8 です。Enter キーをおすと結果を表示します。

- ④ "何かキーを押してください"と表示され、何かキーをおすと、元の画面にもどります。

ランタイムの実行
f・5 キー

? 3, 8
3 * 8 = 24

何かキーを押してください

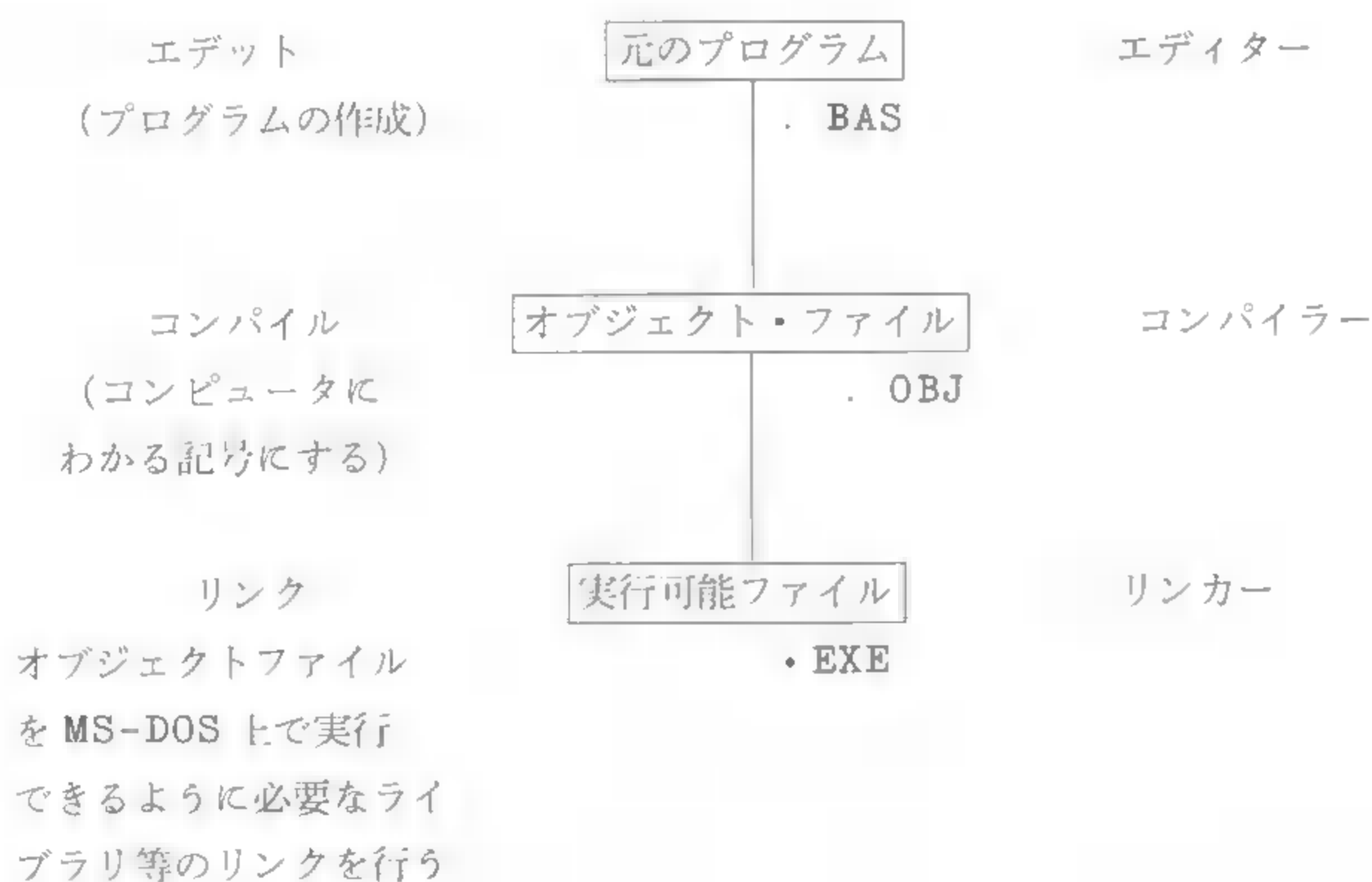
- ⑤ 同じことを実行メニューから S/スタートを選ぶと、項③と同じことになります。

【例題 13】 EXE ファイルの作成

【例題 10】のプログラムを使って EXE ファイルをつくれ。

【解 説】 ◎ EXE ファイルをつくります。

- ① 【例題 10】でつくったプログラムは【例題 11】で `renshu-1.BAS` として保存されます。このファイルをソースファイルと呼びます。BASIC でつくられたファイルですから拡張子に `BAS` がつきます。
- ② 次にこれをコンパイルしてオブジェクトファイルをつくります。この作業を行う機能を持つソフトウェアをコンパイラと呼びます。できたファイルは `OBJ` の拡張子がつきます。
- ③ 次にオブジェクトファイルをリンクして実行可能なファイルをつくります。これを実行可能ファイルあるいは `EXE` ファイルと呼びます。この作業を行う機能を持つソフトをリンカーと呼びます。`EXE` ファイルは拡張子として `EXE` がつきます。
- ④ この `EXE` ファイルが `MS-DOS` 上で実行できるファイルとなります。これらのすべての機能が `Quick BASIC` には備わっています。この環境を称して統合環境といいます。プログラム作成の一連の流れは次のとおりです。



- ⑤ では手順を追っていきましょう。まず、プログラムが画面に表示されているものとします。
- ⑥ 実行メニューを選択します。
- ⑦ `X/EXE ファイル作成...`を選択します。次の画面となります。この画面ではコンパイルスイッチ (バージョン 4.2 では作成形式) を決めます。今回は `X/ランタイム分離型` の前の () 内に * マークがありますので ☒ します。ランタイム分離型 `EXE` ファイルをつくります。

名前を変えたいときは `N/実行ファイル名:` (バージョン 4.2 では `N/EXE ファイル名:`) にカーソルがあるときその名前を入れます。たとえば `RR.EXE` あるいは `RR` です。拡張子 `EXE` は自動的につけられます。 ☒ をすると実行を始めます。

[TAB] キーを使うとカーソルが項目間を移動しますので指定をします。たとえばコンパイルス

イッチ (バージョン 4.2 では作成形式) では(*)X/ランタイム分離型にカーソルが移りますので、
 キーで()A/独立型を指定することができます。M/実行ファイル作成 (バージョン 4.2 では M/EXE ファイル作成) を選んで すれば作成に入ります。E/実行ファイル作成後終了 (バージョン 4.2 では E/EXE ファイル作成後終了) では EXE ファイルをつくってから MS-DOS へもどります。取消で では元の画面にもどります。

[Ver.4.5]

```

F/ファイル  E/編集  V/表示  S/検索  R/実行  D/デバッグ  C/関数  O/オプション  H/ヘルプ
RENSHU-1.BAS

INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
END

```

X/実行ファイル作成
実行ファイル名とコンパイルスイッチを設定してください

N/実行ファイル名: b:renshu-1.exe

[] D/デバッグ **(*) X/ランタイム分離型**
 () A/独立型

< M/実行ファイル作成 > < E/実行ファイル作成後終了 >
 < 取消 > < H/ヘルプ >

⑧ コンパイルとリンクをして、EXE ファイルができます。終了すると元の画面へもどります。

【例題 14】 ファイルの確認

【例題 13】でつくったファイルを MS-DOS 上で確認せよ。

【解 説】 ◎ .EXE ファイルを確認します。

- ① 【例題 13】でつくった EXE ファイルを確認します。EXE ファイルは MS-DOS 上で動くファイルです。
- ② Quick BASIC が使われている状態の場合は MS-DOS へもどして確認します。Quick BASIC では MS-DOS へもどす方法は 2 つあります。第 1 の方法はファイルメニューから X/終了を選びます。MS-DOS の A>となります。Quick BASIC にもどすには"QB"によります。第 2 の方法は一時的に MS-DOS へもどす方法で、ファイルメニューから D/MS-DOS コマンドを選び一時的に MS-DOS のコマンドを使えるようにします。Quick BASIC には Exit と何かキーをおすともどります。
- ③ ここではファイルメニューから D/MS-DOS コマンドで MS-DOS コマンドを使いましょう。
- ④ MS-DOS のコマンドを利用して次のようにします。

DIR によってカレントドライブのファイルのリストができます。DIR/W 内で画面いっぱいになりリストがでます。

DIR B: で B ドライブのファイルのリストがでます。DIR B:/W で B ドライブのリストが画面いっぱいになります。B:のかわりに A:とすれば A ドライブの内容です。

カレントドライブとは MS-DOS が現在使用しているドライブで A>のときは A ドライブ、B>のときは B ドライブです。A>で B: とすれば A ドライブから B ドライブへカレントドライブを変更できます。

- ⑤ Renshu-1.EXE と Renshu-1.OBJ, Renshu-1.BAS があることを確認して下さい。

【例題 15】 MS-DOS で EXE ファイルの実行

【例題 13】でつくったファイル Renshu-1.EXE を実行せよ。

【解 説】 ◎ EXE ファイルを実行します。

- ① Quick BASIC 画面のファイルメニューから M/MS-DOS コマンドを選んで MS-DOS コマンドへ入ります。

renshu-1 により renshu-1.EXE ファイルが実行されます。

- ② 数値を 2 つ入力すると値が求められます。
 ③ EXIT , および何かキーをおすと Quick BASIC へもどります。
 ④ MS-DOS 上では renshu-1 で同様の実行ができます。

MS-DOS コマンドモードでの実行
 実行後 Quick BASIC へもどる画面

```
A>b:renshu-1
? 56,34
56 * 34 = 1904
```

実行
 56 と 34 を入力

EXIT で QuickBASIC に戻ります

Command バージョン 3.30

```
A>exit
```

Quick BASIC へもどる

MS-DOS での実行

```
A>b:renshu-1
? 56,34
56 * 34 = 1904
```

実行
 56 と 34 を入力

```
A>
```

Quick BASIC へもどるには QB とする。

【例題 16】 終 了

【例題 10】のプログラムをつくり実行し、DOS へもどれ。

【解 説】 ◎ DOS へもどります。

- ① ファイルメニューから”X/終了”を選びます。

Quick BASIC を終了し DOS にもどります。Quick BASIC へもどるには QB  から始めます。

3 章 プログラムの読み込み・修正・保存・サブルーチンとファンクションの作成・印刷

【例題 17】 ファイルの呼び出し

ファイル renshu-1.BAS をディスクから呼び出せ.

【解 説】 ① ファイルをディスクから呼び出します.

① ファイルメニューを選びます.

② O/読込を選びます. 次の画面となります.

[Ver.4.5]

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 O/オプション H/ヘルプ
Untitled
O/読込

ファイル名を設定してください

N/ファイル名: *.BAS

ファイル名をかく. B ドライブにしようときは B:ファイル名とする.


A:¥

F/ファイル


D/ディレクトリ

BIN
LIB
[-A-]
[-B-]

< 確認 > < 取消 > < H/ヘルプ >

③ ファイル名に B:Renshu-1.BAS とタイプして  します. B ドライブより Renshu-1.BAS を呼び出します. ドライブを指定するときは a:Renshu-1, b:Renshu-1 です. BAS は省略できます. 大文字, 小文字の区分はありません.

N/ファイル名: renshu-1

④ ファイルがすでにある場合は, **TAB** キーをおすとファイルリストの項目へカーソルが移りますので, 矢印キーを使って希望のファイル名をさがしてカーソルをおき  しても同じです.

⑤ ファイルがないと「ファイルが見つかりません」の表示がされます.  キーで元へもどります.

【例題 18】 プログラムの修正

【例題 10】のプログラムを修正して x 、 y 、 z を入力して、積を求めて結果を表示するプログラムをつくれ。ただし、積は変数 zz に入れる。

【解 説】 ㊦ プログラムの修正をします。

- ① 【例題 17】の方法でファイル Renshu-1.BAS を呼び出します。これを元にして、プログラムの変更をします。

```
INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
END
```

- ② 1 行目の y の後に、 z を追加します。

カーソルを y の後におきます。、と z をタイプします。␣はおしません。 z をタイプした後␣をおすと次に 1 行空白行ができます。⌫キーをおせば元にもどります。

- ③ 次に $z=x*y$ の z を zz とします。

カーソルを z または z の後におき z とタイプします。 $z=x*y$ が $zz=x*y$ となります。Quick BASIC の初期状態では挿入モードとなっていますので z にカーソルをおき z をおせば z が挿入されます。たとえば abc の b にカーソルをおき xyz とタイプすると $axyzbc$ となります。

挿入モードでないとき z とすると z が消えて z がかけられます。 abc で b にカーソルをおき xy とすると axy となります。挿入モードはカーソルをよくみると $_$ (アンダースコア) の点滅となっています。挿入モードでないときはカーソルが四角の反転で点滅となっています。挿入モードは ⌵キーをおすことで切りかえられます。

- ④ $zz=x*y$ の後に $*z$ を追加します。 y の後にカーソルをおき $*z$ とします。␣はしません。
⑤ PRINT 文を修正します。次の①、②、③の手順とします。

```
PRINT x; "*"; y; "="; z
```

③カーソルを ␣ または ␣ キーで移す。
②カーソルを移し z とする。
①カーソル、挿入モード
 $*";z;$ とする。␣はおしません。␣をおすと"以降が次行へうつります。
⌫キーをおすと元へもどります。

- ⑥ 文字の消去は ⌫キーでカーソルのある文字、⌫キーでカーソルの前の文字が消えます。
⑦ これでプログラムができました。ファイルの保存は【例題 19】で行います。

```
INPUT x, y, z
zz = x * y * z
PRINT x; "*"; y; "*"; z; "="; zz
END
```

【例題 19】 同名でファイルを保存

【例題 18】で修正してつくったプログラムを RENSU-1.BAS のまま保存せよ。

【解 説】 ◎ 同名でファイルを保存します。

- ① ファイルメニューから "S/保存" を選びます。

修正されたプログラムが現在のファイル名 RENSU-1.BAS で保存されます。確認の問合わせはありませんから注意してください。修正前のプログラムはなくなります。

【例題 20】 サブルーチンの作成 (1)

次のプログラムをつくれ。このプログラムは x と y を入力して $x^2 + y^2$ を求め、その値をサブルーチン a へ引き渡して表示するものです。

```
DECLARE SUB a (x)
INPUT x, y
z = x * x + y * y
CALL a(z)
END
```

```
SUB a (x)
PRINT x
END SUB
```

(実行例)

```
? 2,3
13
```

2, 3 を入力した例

【解 説】 ◎ サブルーチンをつくります。

- ① まずメインルーチンを次のようにかきます。DECLARE コマンドはファイルへしもうとき自動的にかけられますので、かく必要はありません。

```
INPUT x, y
z = x * x + y * y
CALL a(z)
END
```

- ② 次にサブルーチンをつくります。

編集メニューを選び "S/新規 SUB..." を選びます。

- ③ 続いて、SUB の名前を聞いてきます。   とします。

[Ver.4.5]

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 O/オプション H/ヘルプ
 Untitled

```
INPUT x, y
Z = x * x + y * y
CALL a(z)
END
```

S/新 SUB
 作成するプログラムの名前を入力してください

N/名前: a

< 確認 > < 取消 > < H/ヘルプ >

- ④ 次のように SUB を表示します。

```
SUB a
END SUB
```

- ⑤ SUB a の後に仮引数の (x) をかきます。次行には、PRINT x をかきます。サブルーチンができました。

```
SUB a (x)
PRINT x
END SUB
```

- ⑥ なお、**[SHIFT]** + **[F2]** をおすことでメインルーチンとサブルーチンの間を交互にみることができます。
- ⑦ メインルーチンの x とサブルーチンの x は各々独立しています。これは後ほど詳しく説明します。

【例題 21】 サブルーチンの作成 (2)

次のサブルーチンを含むプログラムをつくれ。

```
DECLARE SUB a (x!)
INPUT x, y
Z = x * y
CALL a(z)
END
```

```
SUB a (x)
PRINT x
END SUB
```

【解 説】 ◎ サブルーチンを含むプログラムをつくります。

- ① Quick BASIC ではサブルーチン SUB, ファンクション FUNCTION を使うことができます。

SUB を例にサブルーチンを含むプログラムのつくり方を示します。なおプログラムの使い方については I 編 20 章で詳しく扱います。

- ② 次のようにプログラムをかきます。

```
INPUT x, y
z = x * y
CALL a(z)
END
```

- ③ 次に最後の行に SUB a(x) とかきます。

```
INPUT x, y
z = x * y
CALL a(z)
END
sub a(x)
```

- ④  またはカーソルを別の行へ移すと次のようにサブルーチン画面となります。

```
SUB a (x)
END SUB
```

- ⑤ カーソルを中間のスペース行におき、PRINT 文をかきます。

```
SUB a (x)
PRINT x
END SUB
```

- ⑥ これでプログラムはできあがりです。先頭の "DECLARE SUB a(x!)" は自動的につけられます。
 ⑦ メインプログラムの画面とサブルーチンの画面の切りかえは **SHIFT** + **f.2** キーで行います。
 ⑧ これを RRR-95 の名前で保存します。今回はテキストファイルで保存します。

【例題 22】 モジュールの追加

【例題 21】のプログラム（ファイル名 RRR-95）を修正して次のプログラムとせよ。ただし、RRR-95 のプログラムを追加してから修正していくものとする。

プログラム例

```
DECLARE SUB a1 (z1!)
DECLARE SUB a (x!)
INPUT x, y
z = x * y
CALL a(z)
INPUT x1, y1
z1 = x1 / y1
CALL a1(z1)
END
```

```
SUB a (x)
PRINT x
END SUB
```

```
SUB a1 (z1)
PRINT z1
END SUB
```

実行結果例



```
? 5,3      5 と 3, 2 と 3 を入力の場合
15
? 2,3
.66666667
```

【解 説】 ◎ ファイルの追加をします。

- ① 【例題 21】のプログラムの"END"の部分にカーソルをおきます。
- ② ファイルメニューを選択します。次に"M/結合..." (バージョン 4.5), M/追加..." (バージョン 4.2) を選びます。
- ③ 追加するファイル名を問合わせます。RRR-95 を指定します。

N/ファイル名: b:rrr-95

B:¥

- ④  により次のように RRR-95 の内容が追加されます。なお、ここではサブルーチンがすでに存在するものと同名になりますので図のような警告がでますが、修正しますのでこれは無視して  をおします。[ESC] キーをおします。メインプログラムの画面となります。

Ver.4.2

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 H/ヘルプ (f.1)
RRR-95.BAS

```
DECLARE SUB ■ (x!)
INPUT x, y
z = x * y
CALL a(z)
DECLARE SUB a (x!)
INPUT x, y
z = x * y
CALL a(z)
END
```

追加ファイル

同名のサブプログラムを読み込もうとしました
a

警告
同名のサブルーチン
が存在する

```
SUB a (x)
PRINT x
END SUB
```

確認

END

Ver.4.5

同名のプロシージャを読み込もうとしました
a

警告
同名のプロシージャが
存在する

< 確認 > < H/ヘルプ >

- ⑤ バージョン 4.5 とバージョン 4.2 では若干結合・追加のされ方が異なりますが、いずれも表示を利用して、問題のプログラムのよう修正します。

Ver.4.5

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 O/オプション H/ヘルプ
RRR-95.BAS

```
DECLARE SUB a (x!)
INPUT x, y
z = x * y
CALL a(z)
END
```

```
DECLARE SUB a (x!)
INPUT x, y
z = x * y
CALL a(z)
END
```

```
SUB a (x)
PRINT x
END SUB
```

【例題 23】 ファンクション

次のプログラムをつくれ。このプログラムは x を入力し、ファンクション f に x を引きわたし、その 3 乗を求めて表示するものとする。

```
DECLARE FUNCTION f! (z!)
INPUT x
PRINT f(x)
END
```

```
? 3      3 を入力の例
      27
? 10     10 を入力の例
      1000
```

```
FUNCTION f (z)
f = z * z * z
END FUNCTION
```

【解 説】 ㊦ ファンクションをつくります。

- ① 問題のメインルーチンをかきます。DECLARE コマンドは自動的につけられますからかく必要はありません。FUNCTION については 21 章で詳しく学びます。
- ② 編集メニューを選択します。ここで "F/新規 FUNCTION ..." を選びます。
- ③ 次の画面となります。名前を入力します。

名前: f

- ④  により次の画面となります。

```
FUNCTION f
END FUNCTION
```

- ⑤ FUNCTION を次のようにかきます。これでできあがりです。

```
FUNCTION f (z)
f = z * z * z
END FUNCTION
```

【例題 24】 ファイルの印刷

【例題 21】のファイルを印刷するプログラムをつくれ。

【解 説】 ㊦ ファイルの印刷をします。

- ① 【例題 21】のプログラムが表示されているものとします。
- ② ファイルメニューから P/印刷 ... を選びます。次の画面となります。

[Ver.4.5]

P/印刷
印刷する範囲を選択してください

() S/指定範囲
() W/現在のウィンドウ
(*) M/現在のモジュール
() A/全体

< 確認 > < 取消 > < H/ヘルプ >

[Ver.4.2]

印刷
() S/指定範囲
() W/アクティブ・ウィンドウ
(*) M/カレント・モジュール
() A/全体

確認 取消

- ③ 印刷の種類を 4 つの中で選びます。矢印キーを使います。既定値は M/現在のモジュールでアクティブウインドウのモジュールです。サブルーチンがあればサブルーチンも印刷します。S/指定範囲はカーソルのある行を印刷します。W/アクティブウインドウはアクティブウインドウ全体です。つまり、現在表示されているウインドウのみです。A/全体はメモリーのモジュール、インクルードファイル、ドキュメントファイルすべてを印刷します。

【結 果】

M/カレントモジュールの例
 アクティブウインドウ

```

DECLARE SUB a (x!)
INPUT x, y
z = x * y
CALL a(z)
END

```

SABルーチン

```

SUB a (x)
PRINT x
END SUB

```

A/全体の例
 アクティブウインドウ

```

DECLARE SUB a (x!)
INPUT x, y
z = x * y
CALL a(z)
END

```

SABルーチン

```

SUB a (x)
PRINT x
END SUB

```

W/アクティブウインドウの例
 アクティブウインドウ (サブルーチンは印刷しない)

```

INPUT x, y
z = x * y
CALL a(z)
END

```

W/アクティブウインドウの例
 アクティブウインドウ (メインプログラムは印刷しない)
 [SHIFT] + [F2] でメインプログラムとサブルーチンのウインドウを切りかえる。

```

SUB a (x)
PRINT x
END SUB

```

S/指定範囲の例
 z にカーソルをおいた場合

```

z = x * y

```


4章 ダイレクト実行

【例題 25】 ダイレクトモード

x 度を入力して、 $12 * \text{SIN}(x \text{ 度})$ を求めて表示するプログラムをつくり、実行前に $\text{SIN}(15 \text{ 度})$ の値をダイレクトモードで確認せよ。

プログラム例

```
INPUT x
PRINT 12 * SIN(x / 180 * 3.14159)
END
```

15 を入力の結果例

```
? 15
3.105826
```

【解 説】 ① ダイレクトモードで直接命令を実行します。

① プログラムをつくった後、**(F-6)** によりダイレクトモードをアクティブ画面にします。

② ダイレクト画面に次のように書き込みます。

```
INPUT x
PRINT 12 * SIN(x / 180 * 3.14159)
END
```

ダイレクト モード

```
print sin (15/180*3.14159)
```

③ **(J)** により 15 度の SIN の値を求めることができます。

```
.2588189
```

【例題 26】 ダイレクトモードで確認してからプログラムへ挿入

【例題 25】と同じプログラムをつくれ。ただし、2 行目の PRINT 文は、ダイレクトモードで 15 度の SIN を求める命令を書き実行して、確認してから、挿入して修正してプログラムを完成するものとする。

【解 説】 ◎ ダイレクトモードで命令を確認してからプログラムを挿入します。

- ① 次のように 2 行を記入したところで、**[F6]** によりダイレクトモードをアクティブ画面とします。
ここで 15 度の SIN を求める式をかき実行し確認します。

```
INPUT x
END
```

```
print sin (15/180*3.14159)
```

ダイレクト モード

- ② "print sin(15/180*3.14159)"にカーソルをおき、**[CTRL] + [Y]**をおします。この命令がクリップボードにコピーされます。編集メニューで P/ペーストを選ぶと再びこの命令がカーソルのあるダイレクトモードの 1 行目にかかれます。**[F6]** によりアクティブ画面をかえ "END" にカーソルを移し、編集メニューから P/ペーストを選んでこの命令を "END" の前に挿入します。

```
INPUT x
PRINT SIN(15 / 180 * 3.14159)
END
```

```
print sin (15/180*3.14159)
```

ダイレクト モード

- ③ "12*"の挿入と 15 を x に変更してプログラムを完成します。

```
INPUT x
PRINT 12 * SIN(x / 180 * 3.14159)
END
```

```
print sin (15/180*3.14159)
```

ダイレクト モード

【例題 27】 ダイレクトモードでサブルーチンの実行

次のプログラムをつくり、次にサブルーチン a1, a2, a3 を各々独立して実行せよ。このプログラムは x 度を入力して、x 度の sin, cos, tan をサブルーチンを使って求めるものです。

```

DECLARE SUB a1 (a1)
DECLARE SUB a2 (a2)
DECLARE SUB a3 (a3)
INPUT x
CALL a1(x)
CALL a2(x)
CALL a3(x)
END

SUB a1 (z1)
  PRINT SIN(z1 / 180 * 3.14159)
END SUB

SUB a2 (z2)
  PRINT COS(z2 / 180 * 3.14159)
END SUB

SUB a3 (z3)
  PRINT TAN(z3 / 180 * 3.14159)
END SUB

```

```

? 25          25 を入力した結果例
.422618
.9063079
.4663073

```

【解 説】 ① サブルーチンを独立してダイレクトに実行します。

① プログラムをかきます。

② **f・6** でアクティブ画面をダイレクトモードとします。

次のように call 命令をかきます。 **F6** によりサブルーチンのみ実行します。

CALL a1(30) **F6** で 30 度の sin の値を求めます。

CALL a2(75) **F6** で 75 度の cos の値を求めます。

CALL a3(45) **F6** で 45 度の tan の値を求めます。

```

INPUT x
CALL a1(x)
CALL a2(x)
CALL a3(x)
END

```

ダイレクト モード

```

call a1(30)
call a2(75)
call a3(45)

```

ダイレクト実行の結果

```
.4999996
.2588201
.9999988
```

【例題 28】 変数の値を変えて実行

1 から 1000 までを表示するプログラムをつくり、途中 **STOP** キーでプログラムを中断し、変数の値を変えて、そこから再開して 1000 まで表示せよ。

【解 説】 ◎ 変数の値をダイレクトモードで変更します。

- ① 次のプログラムをつくり、**f・5** で実行します。次に **STOP** キーをおして途中で中断します。ここで **f・6** キーでアクティブ画面をダイレクトモードとして、**i=950** **↓** とします。中断したときの **i** の値に関係なく **i** は 950 となります。

```
FOR i = 1 TO 1000
  PRINT i;
NEXT i
PRINT
END
```

ダイレクトモード

i=950

- ② **f・5** で再開します。i が 1000 まで実行されてプログラムが終了します。
- ③ 次の例はプログラムを実行し、**STOP** キーをおしたとき **i** は 179 でそれを表示したあと **NEXT i** の実行前で止まりました。"NEXT i" が緑色になっています。i=950 をダイレクト実行して再開しましたので、NEXT i により **i** は 951 となり 951 から 1000 まで表示します。

【結 果】

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42		
43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62		
63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82		
83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	10		
2	103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	11					
8	119	120	121	122	123	124	125	126	127	128	129	130	131	132	133	13					
4	135	136	137	138	139	140	141	142	143	144	145	146	147	148	149	15					
0	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	16					
6	167	168	169	170	171	172	173	174	175	176	177	178	179	951	952	95					
3	954	955	956	957	958	959	960	961	962	963	964	965	966	967	968	96					
9	970	971	972	973	974	975	976	977	978	979	980	981	982	983	984	98					
5	986	987	988	989	990	991	992	993	994	995	996	997	998	999	1000						

5 章 サブファイル・テキストファイル・メインプログラムの設定・インクルードファイル

【例題 29】 サブファイル

次の 2 つのファイルをつくり、各々、RRR-71.BAS, RRR-72.BAS で保存する。次に、RRR-71 のサブルーチン plus と RRR-72 のサブルーチン multi を呼び出して、入力した値までの 1 からの和と階乗を求めるプログラム RRR-73.BAS をつくれ。なお、RRR-71 は値を入力して、1 からその値までの和を求めるプログラム、RRR-72 は値を入力して、階乗を求めるプログラムとする。

RRR-71.BAS のプログラムと
6 を入力の結果

```
DECLARE SUB plus (z%, zz%)
INPUT x%
CALL plus(x%, y%)
PRINT "1 から "; x%; "マデ`ノ 7="; y%
END
```

```
SUB plus (z1%, z2%)
FOR i = 1 TO z1%
    y% = y% + i
NEXT i
z2% = y%
END SUB
```

```
? 6
1 から 6 マデ`ノ 7= 21
```

RRR-72.BAS のプログラムと
6 を入力の結果

```
DECLARE SUB multi (z%, zz%)
INPUT x%
CALL multi(x%, yy%)
PRINT "1 から "; x%; "マデ`ノ カイシ`ヨウ="; yy%
END
```

```
SUB multi (z1%, z2%)
yy% = 1
FOR i = 1 TO z1%
    yy% = yy% * i
NEXT i
z2% = yy%
END SUB
```

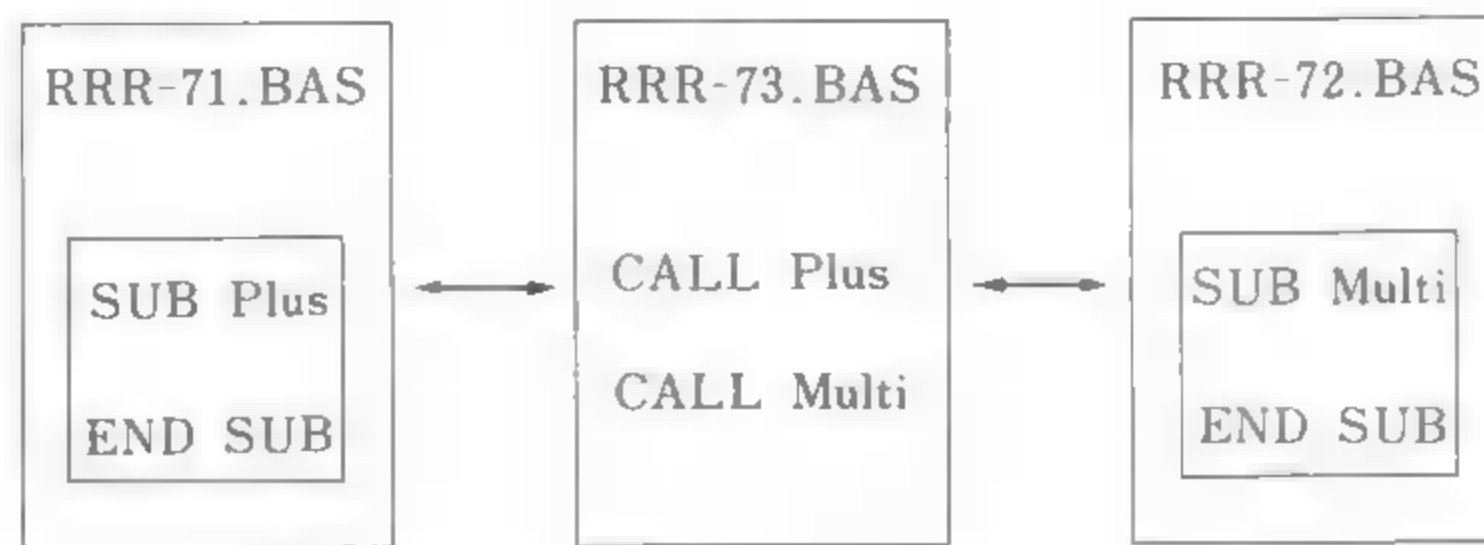
```
? 6
1 から 6 マデ`ノ カイシ`ヨウ= 720
```

RRR-73.BAS のプログラム

```
INPUT x%
CALL plus(x%, y%)
CALL multi(x%, yy%)
PRINT y%
PRINT yy%
END
```

【解 説】 ◎ サブファイルを利用します。

- ① Quick BASICでは複数のファイルをロードして、別々に管理するとともに、別のファイルのモジュールをメインモジュールから利用することができます。今回の例は次のようなものです。



- ② RRR-71.BAS, RRR-72.BAS, RRR-73.BAS を各々作成し、保存します。
- ③ ファイルメニューから“L/サブファイル読込... (バージョン 4.5)”, L/サブファイル常駐... (バージョン 4.2) を選びます。
- ④ ファイル名を問合わせてきますので rrr-71 を入力します。なお形式は“M/モジュール”とします。

[Ver.4.5]

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 O/オプション H/ヘルプ

RRR-71.BAS
L/サブファイル読込

DECL
INPU ファイル名、ファイル形式を設定してください
CALL
PRIN N/ファイル名: **rrr-71**
END

B:¥

F/ファイル		D/ディレクトリ
RENSHU-1.BAS	RRR-95.BAS	[-A-]
RENSHU-2.BAS	RRR-96.BAS	[-B-]
RRR-100.BAS	RRR-97.BAS	
RRR-71.BAS	RRR-98.BAS	
RRR-72.BAS	RRR-99.BAS	
RRR-73.BAS		

(*) **M/モジュールプログラム**
() I/インクルードファイル
() D/ドキュメントファイル

< 確認 > < 取消 > < H/ヘルプ >

- ⑤ 同様に、ファイルメニューから **L** , rrr-72 **J** , サブファイルから **L** , rrr-73 **J** により 3つのファイルをメモリへ呼びます。
- ⑥ 実行メニューから“M/メインモジュールの設定...”を選びます。
- ⑦ メインプログラムの設定画面となります。ここでは実行するのは rrr-73 ですから、カーソルを rrr-73 へ移し、**J** します。rrr-73 がメインプログラムとなります。

[Ver.4.5]

M/メインモジュールの設定
メインモジュールを選択してください

```
RRR-73.BAS
RRR-71.BAS
RRR-72.BAS
```

[Ver.4.2]

メインプログラムの選択:

```
RRR-71.BAS
RRR-72.BAS
RRR-73.BAS
```

- ⑧ **f.2** をおすと次のようにメインプログラムが rrr-73 と確認できます.

[Ver.4.5]

S/SUB一覧
C/編集するプロシージャ、ウィンドウを選択してください

```
RRR-73.BAS
RRR-71.BAS
plus
RRR-72.BAS
multi
```

RRR-73.BAS:=メインモジュール

[Ver.4.2]

C/プログラム選択:

```
RRR-73.BAS
RRR-71.BAS
plus
RRR-72.BAS
multi
```

RRR-73.BAS:=メインプログラム

- ⑨ **f.5** で実行します. rrr-73 のメインプログラムが, rrr-71 のサブルーチン plus と rrr-72 のサブルーチン multi を使って結果を求めます.

【結 果】

```
? 6      (6 を入力の場合)
21
720
```

【例題 30】 サブファイルの開放

【例題 29】の状態から rrr-71, rrr-72 を開放せよ.

【解 説】 ◎ サブファイルを開放します.

- ① 【例題 29】の手順で, サブファイル rrr-71, rrr-72 をメモリへ読み込みます.

- ② ファイルメニューを選びます。
- ③ "U/サブファイル開放..."を選びます。メモリに常駐しているファイル名が一覧されます。まず、`rrr-72.BAS`にカーソルを移し **[J]** します。

[Ver.4.5]

[Ver.4.2]

U/サブファイル解放
解放するモジュールを選択してください

RRR-73.BAS
RRR-71.BAS
RRR-72.BAS

解放するモジュールの選択:

RRR-73.BAS
RRR-71.BAS
RRR-72.BAS

- ④ `rrr-72.BAS` がなくなります。これは **[f.2]** キーで確認できます。

C/プログラム選択:

RRR-73.BAS
RRR-71.BAS
plus

- ⑤ 同様にファイルメニューから **[U]** を行い、ファイル名に `rrr-71.BAS` を指定して **[J]** します。
`rrr-71.BAS` がメモリから消えます。`rrr-73.BAS` のみ残ります。

【例題 31】 メインプログラムの設定

次の3つのプログラムを各々`rrr-74`、`rrr-75`、`rrr-76`で保存し、次に各々をメモリに呼んで、`rrr-75`、`rrr-74`、`rrr-76`の順で実行せよ。

RRR-74	INPUT x% PRINT x% END
RRR-75	INPUT x% PRINT x% * x% END
RRR-76	INPUT x% PRINT x% * x% * x% END

【解 説】 ㉟ 複数プログラムを読み込んで任意のプログラムを実行します。

- ① `rrr-74`、`rrr-75`、`rrr-76`のプログラムをつくり各々保存します。
- ② ファイルメニューを選択します。
- ③ "L/サブファイル読込..." (バージョン 4.5) または "L/サブファイル常駐..." (バージョン 4.2) を選びます。ファイル名を問合わせてきますので、`rrr-74` **[J]** とします。このとき、形式は M/モジュールとします。ファイル `rrr-74` がメモリに読み込まれます。
- ④ 同様に `rrr-75` と `rrr-76` をメモリに読み込みます。
- ⑤ **[f.2]** をおすと読み込まれたファイル名がリストされます。

C/プログラム選択:

RRR-74.BAS
RRR-75.BAS
RRR-76.BAS

- ⑥ まず `rrr-75` を実行します。
実行メニューを選びます。
"M/メインモジュールの設定 ..." を選びます。
- ⑦ "`rrr-75.BAS`" を選択して **[J]** します。 `rrr-75` がメインプログラムとなります。
- ⑧ **[f.5]** で `rrr-75` が実行されます。
- ⑨ 同様に実行メニューから M/メインモジュールの設定 ... を選び、 `rrr-76` **[J]** により `rrr-76` をメインプログラムとして **[f.5]** で `rrr-76` を実行します。
- ⑩ 同様に実行メニューから "M/メインモジュールの設定 ..." を選び、 `rrr-74` **[J]** として **[f.5]** で `rrr-74` を実行します。

【結 果】

```

? 3
 9
? 4
64
? 8
 8

```

【例題 32】 マルチモジュールプログラムの保存

【例題 31】 でつくった `RRR-74`、`RRR-75`、`RRR-76` の各プログラムを一緒に保存せよ。

【解 説】 ㉠ マルチモジュールプログラムを保存します。

- ① `RRR-74`、`RRR-75`、`RRR-76` の 3 つのプログラムを同時に保存することができます。
- ② サブファイル常駐により 3 つのプログラムがメモリにあるものとします。ファイルメニューを選択します。
"V/すべて保存" を選びます。3 つのプログラムが同時に保存されます。
- ③ ファイル名はメインモジュール（今回は `rrr-74.BAS`）の `rrr-74` 拡張子を `.MAK` とするメインファイルをつくります。

【例題 33】 マルチモジュールプログラムの読み込み

【例題 32】 で保存したマルチモジュールプログラムを読み込め。

【解 説】 ㉡ マルチモジュールプログラムを読み込みます。

- ① ファイルメニューを選択します。
- ② "O/読込 ..." を選びます。
- ③ ファイル名に `RRR-74.MAK` を入力します。3 つのモジュールからなるマルチプログラムが読み込まれます。
- ④ 確認のため **[f.2]** キーをおすと、次のように 3 つのプログラム名がかかれています。

```

RRR-74.BAS
RRR-75.BAS
RRR-76.BAS

```

【例題 34】 ドキュメントファイルの作成

次のドキュメントをつくりファイルワープロ 1.として保存せよ。

拝啓 ますますご清栄のこととお喜び申し上げます
 さて、下記の通り定例会を開催しますので
 ご出席ください

敬具

記

1. 日時 :
2. 場所 :
3. 議題 :
4. 出席者 :
5. その他 :

以上

【解 説】 ㉟ ドキュメントファイルをつくります。

- ① ファイルメニューを選びます。
- ② "C/サブファイル作成 ..." を選びます。
- ③ 次のファイル名入力画面となります。

[Ver.4.5]

C/サブファイル作成
 ファイル名、ファイル形式を設定してください

N/ファイル名:

- (*) M/モジュールプログラム
- () I/インクルードファイル
- () D/ドキュメントファイル

＜ 確 認 ＞ ＜ 取 消 ＞ ＜ H/ヘルプ ＞

[Ver.4.2]

N/ファイル名:

形 式

- (*) M/モジュール
- () I/インクルード
- () D/ドキュメント

- ④ ファイル名を"ワープロ 1.", "D/ドキュメント"を選択します。通常のプログラムは M/モジュールによって、構文チェックが行われます。ドキュメントファイルでは BASIC 文を使うわけではありませんから D/ドキュメントによって、構文チェックをしないようにします。

[Ver.4.5]

C/サブファイル作成
ファイル名, ファイル形式を設定してください

N/ファイル名: ワ-ア 01

- () M/モジュールプログラム
() I/インクルードファイル
(*) D/ドキュメントファイル

[Ver.4.2]

N/ファイル名: ワ-ア 01

- 形式
() M/モジュール
() I/インクルード
(*) D/ドキュメント

- ⑤ 文章をかき込みます。
⑥ ファイルメニューから [P] で印刷, ファイルメニューから [S] でファイルに保存されます。
⑦ この機能を使って CONFIG.SYS や AUTOEXEC.BAT などを書き換えることができます。

【例題 35】 ドキュメントファイル

【例題 34】のドキュメントファイルを使って, 次のドキュメントをつくれ。

拝啓

ますますご清栄のこととお喜び申し上げます

さて, 下記の通り定例会議を開催しますので
ご出席ください

敬具

1. 日時:	記 11月15日 15時から16時
2. 場所:	第三会議室
3. 議題:	創立30周年行事について
4. 出席者:	総務部全員
5. その他:	前回の宿題を持参のこと

以上

【解 説】 ㉠ ドキュメントファイルの追加をします。

- ① ファイルメニューから [O], ワ-プロ 1. [J] により, 【例題 34】のドキュメントを読み出します。
② カーソルを移して, "11月15日……持参のこと"を追加して記入します。
③ 保存はファイルメニューから [A], ワ-プロ 2. [J], 印刷はファイルメニューから [P] です。

【例題 36】 インクルードファイルの作成

下のインクルードファイルをファイル名"HAIRETSU.IN"でつくれ。

```
DIM a$(20)
COMMON a$()
```

【解 説】 ◎ インクルードファイルの作成

- ① インクルードファイルをつくります。この使い方はⅡ編 24 章で扱います。
- ② ファイルメニューを選びます。次に "C/サブファイル作成 ..." を選びます。
- ③ ファイル名を問合わせますので "hairetsu.in" とします。形式を "I/インクルード" とします。
 により初期画面へもどります。

[Ver.4.5]

C/サブファイル作成
 ファイル名、ファイル形式を設定してください

N/ファイル名: **hairetsu.in**

() M/モジュールプログラム
 (*) I/インクルードファイル
 () D/ドキュメントファイル

< 確認 > < 取消 > < H/ヘルプ >

[Ver.4.2]

N/ファイル名: **hairetsu.bas**

形 式
 () M/モジュール
 (*) I/インクルード
 () D/ドキュメント

確 認

取 消

- ④ 問題の 2 行をかきます。これは配列 A\$(20) を宣言し、これをグローバル変数とするものです。
- ⑤ ファイルメニューを選択し、 により保存を選びます。ファイル名を b:hairetsu.in、形式を "T/テキスト" として します。インクルードファイルがドライブ b に格納されます。

【例題 37】 インクルードファイルの実行と表示

次のプログラムをファイル名 RRR-10.BAS でつくり、次に【例題 36】でつくったインクルードファイルをつないで実行し、次にインクルードファイルを表示せよ。

```
REM $INCLUDE: 'B:HAIRETSU.IN'
FOR i = 0 TO 19
  READ a$(i)
NEXT i
FOR i = 0 TO 19
  IF a$(i) = "Tokyo" THEN PRINT a$(i)
NEXT i
END
DATA Osaka,Nagoya,Ueno,Kyoto
DATA Kobe,Nara,Kagoshima,Sapporo
DATA Yamaguchi,Yokohama,Chiba,Tsu
DATA Fukushima,Tokyo,Sendai,Miyagi
DATA Shizuoka,Hamamatsu,Mishima,Kanazawa
```

【解 説】 ◎ インクルードファイルの連結と表示をします。

- ① 【例題 36】のファイル HAIRETSU.IN を読み込みます。ファイルメニューから で次の画面となります。ファイル名を B:HAIRETSU.IN、I/インクルードファイルを指定して します。

これでインクルードファイルが読み込まれ、ファイル RRR-10 につながります。

```

RRR-10.BAS
L/サブファイル読込
ファイル名、ファイル形式を設定してください
N/ファイル名: b:hairetsu.in
B:¥
F/ファイル
D/ディレクトリ

RENSHU-1.BAS  RRR-51.BAS  RRR-73.BAS  [-A-]
RENSHU-2.BAS  RRR-6.BAS   RRR-74.BAS  [-B-]
RRR-10.BAS    RRR-7.BAS   RRR-75.BAS
RRR-100.BAS   RRR-70.BAS  RRR-76.BAS
RRR-2.BAS     RRR-71.BAS  RRR-85.BAS
RRR-50.BAS    RRR-72.BAS  RRR-95.BAS

( ) M/モジュールプログラム
(*) I/インクルードファイル
( ) D/ドキュメントファイル

< 確認 > < 取消 > < H/ヘルプ >

```

- ② **f-5** で実行します。Tokyo と表示されます。
- ③ 表示メニューから“L/インクルードファイル表示”を選択します。RRR-10 で使われているインクルードファイル **hairetsu.in** が表示されます。
- ④ 次のようにインクルードファイルの実行される部分へ挿入された形で表示されます。

```

RRR-10.BAS
REM $INCLUDE: 'B:HAIRETSU.IN'
DIM a$(20)
COMMON a$()

FOR i = 0 TO 19
  READ a$(i)
NEXT i
FOR i = 0 TO 19
  IF a$(i) = "Tokyo" THEN PRINT a$(i)
NEXT i
END
DATA Osaka,Nagoya,Ueno,Kyoto
DATA Kobe,Nara,Kagoshima,Sapporo
DATA Yamaguchi,Yokohama,Chiba,Tsu
DATA Fukushima,Tokyo,Sendai,Miyagi
DATA Shizuoka,Hamamatsu,Mishima,Kanazawa

```

- ⑤ 項③を再度行くと元へもどります。

【例題 38】 インクルードファイルの編集

【例題 36】のインクルードファイル“hairetsu.in”の修正画面をつくれ。

【解 説】 ◎ インクルードファイルの編集をします。

- ① 表示メニューから“**I/インクルードファイル編集**”を選びます。
- ② インクルードファイル“hairetsu.in”が表示され、編集が可能となります。

```
DIM a$(20)  
COMMON a$()
```

- ③ 編集後はファイルメニューを選択し、**S** によって保存を選び保存します。

6 章 表 示

【例題 39】 画面分割

次のプログラムを使い、画面を分割して、上をメインルーチン、下を PR のサブルーチンの表示とせよ。なおこのプログラムは 1 から 100 の和を求めるものとする。

```
DECLARE SUB pr (z)
t = 0
FOR i = 1 TO 100
    t = t + i
NEXT i
CALL pr(t)
END

SUB pr (z)
    PRINT "1+2+3+.....+100="; z
END SUB
```

【結 果】

1+2+3+.....+100= 5050

【解 説】 ◎ 画面を上下に分割します。

- 1) メインルーチンとサブルーチン PR をつくります。ファイル名 RRR-70 で保存します。
- 2) 表示メニューを選びます。ここで、"P/画面分割"を選びます。分割をもどすときも、この操作です。
- 3) 次の 2 画面となります。上下ともメインルーチンです。

RRR-70.BAS

```
DECLARE SUB pr (z)
t = 0
FOR i = 1 TO 100
    t = t + i
NEXT i
CALL pr(t)
END
```

RRR-70.BAS

```
DECLARE SUB pr (z)
t = 0
FOR i = 1 TO 100
    t = t + i
NEXT i
CALL pr(t)
END
```

ダイレクト モード

- ④ **f・6** によって下側がアクティブウィンドウとなります。**SHIFT** と **f・2** をおすと内容はサブルーチン PR となります。

```

RRR-70.BAS
DECLARE SUB pr (z)
t = 0
FOR i = 1 TO 100
    t = t + i
NEXT i
CALL pr(t)
END

```

```

RRR-70.BAS:pr
SUB pr (z)
PRINT "1+2+3+.....+100="; z
END SUB

```

ダイレクト モード

- ⑤ さらに **f・6** をおすとダイレクトモード、さらに **f・6** をおすと上側がアクティブウィンドウとなります。

【例題 40】 アクティブ画面のフルスクリーン

【例題 39】のプログラムでビューウィンドウを 2 分割し、上段にメインプログラム、下段にサブルーチン PR を表示し、下段をアクティブ画面としてからその画面をフルスクリーンとせよ。

【解 説】 ◎ アクティブ画面をフルスクリーンとします。

- ① 【例題 39】の手順で、ビューウィンドウを 2 分割し、下段をアクティブ画面とし、かつサブルーチンを表示する画面とします。
- ② **CTRL** + **f・10** によりアクティブ画面をフルスクリーンとします。この操作で元へもどります。

```

RRR-70.BAS:pr
SUB pr (z)
PRINT "1+2+3+.....+100="; z
END SUB

```

【例題 41】 アクティブ画面の拡大・縮小

【例題 39】のプログラムを表示した後、ダイレクトモードウィンドウを拡大せよ。

【解 説】 ◎ アクティブ画面の拡大・縮小をします。

- ① 【例題 39】のプログラムをつくります。

RRR-70.BAS

```

DECLARE SUB pr (z)
t = 0
FOR i = 1 TO 100
  t = t + i
NEXT i
CALL pr(t)
END

```

ダイレクト モード

- ② **[F6]** キーによって、ダイレクトモードウィンドウをアクティブ画面とします。
- ③ **[GRPH]** キーと **[+]** キーをおすとそのたびにダイレクトモードウィンドウが拡大します。

RRR-70.BAS

```

DECLARE SUB pr (z)
t = 0
FOR i = 1 TO 100
  t = t + i
NEXT i
CALL pr(t)
END

```

ダイレクト モード

- ④ **[GRPH]** キーと **[-]** キーによって、逆にアクティブウィンドウを縮小できます。

ウィンドウの分割のまとめ

ウィンドウは①ビュー・ウィンドウ、②ダイレクト・モード・ウィンドウ、③ウォッチ・ウィンドウの3種があります。

Ver.4.5

F/ファイル	E/編集	V/表示	S/検索	R/実行	D/デバッグ	C/関数	O/オプション	H/ヘルプ
RRR-70.BAS t: <FALSE>								
RRR-70.BAS i: <FALSE>								
RRR-70.BAS								
<pre> DECLARE SUB pr (z!) t = 0 FOR i = 1 TO 100 t = t + i NEXT i CALL pr(t) END </pre>								
ダイレクト モード								
<SHIFT+F1=ヘルプ> <F6=窓切換> <F2=SUB一覧> <F5=実行> <F8=トレース>								
00001:001								

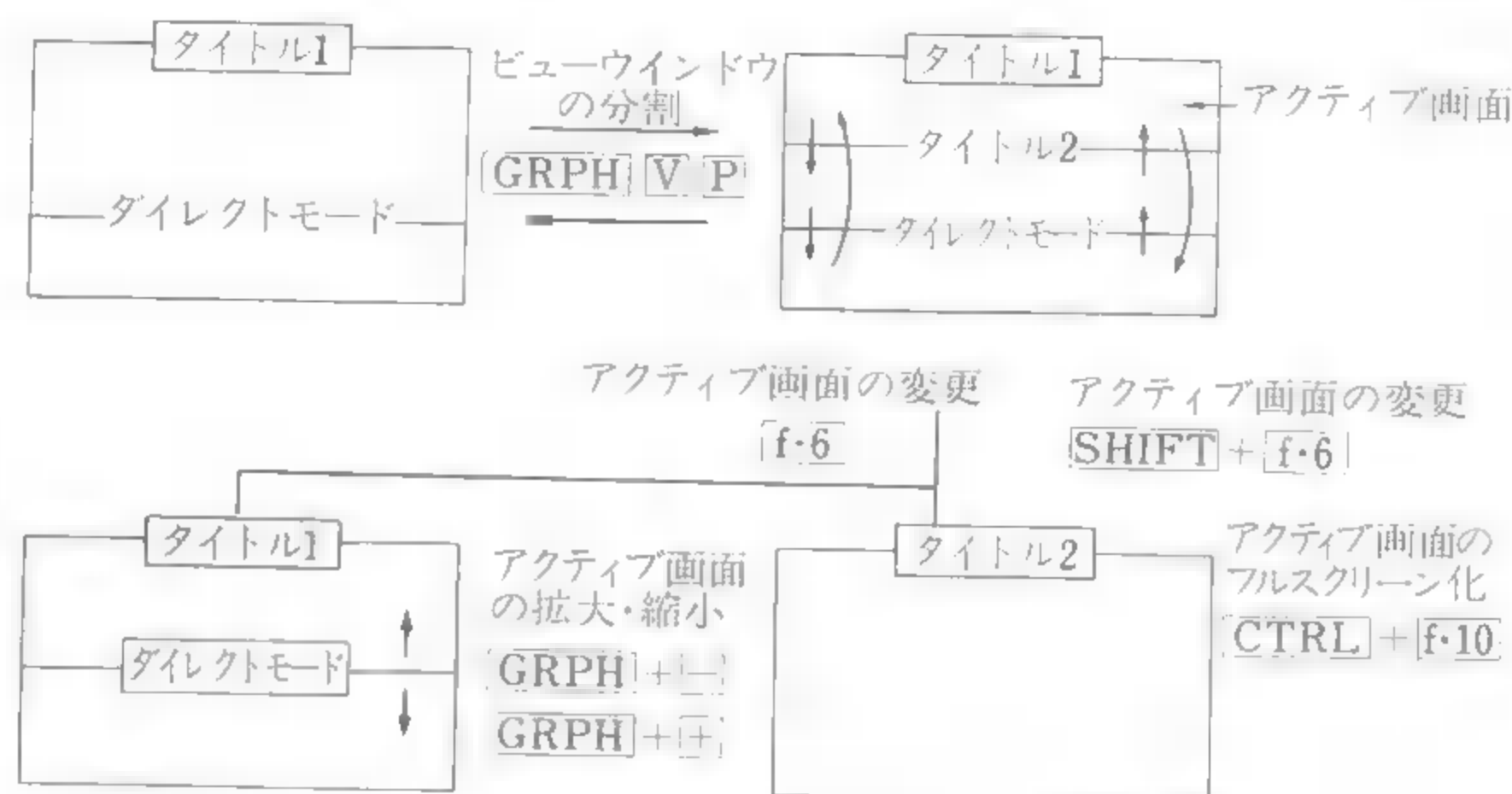
ウォッチ・
ウィンドウ

ビュー・ウ
ィンドウ

ダイレクト
・モード・
ウィンドウ

ビュー・ウィンドウではタイトルが反転表示されている画面がアクティブ画面です。ダイレクト・モード・ウィンドウがアクティブな場合は“ダイレクト・モード”の部分が反転表示されます。

ビュー・ウィンドウは2つに分割できます。分割されたビュー・ウィンドウのどちらかをアクティブにしたり、ダイレクト・モードをアクティブにしたりできます。また、アクティブな画面をフルスクリーンにできます。また、アクティブウィンドウの大きさを拡大したり、縮小したりできます。これらの関係を図に示します。



【例題 42】 サブルーチンの一覧

次のプログラムをつくり、サブルーチンの一覧を確認せよ。ただし、このプログラムはファイル名は RRR-7.BAS とします。

```

DECLARE SUB a1 (z%, zz%, zzz%)
DECLARE SUB a2 (z AS INTEGER, zz AS INTEGER)
INPUT x%
STOP
CALL a1(x%, xx%, xxx%)
PRINT xx%
PRINT xxx%
END

SUB a1 (z%, zz%, zzz%)
zz% = 0
FOR i = 1 TO z%
    zz% = zz% + i
NEXT i
STOP
CALL a2(z%, yy%)
zzz% = yy%
END SUB

SUB a2 (y%, yy%)
z% = 1
FOR i = y% TO 1 STEP -1
    z% = z% * i
NEXT i
yy% = z%
STOP
END SUB

```

【解 説】 ◎ サブルーチンの一覧をつくります。

① 表示メニューを選びます。

- ② ここで「S/SUB 一覧 ...」を選びます。次のようにサブルーチンの一覧ができます。

[Ver.4.5]

V/表示

S/SUB一覧

C/編集するプロシージャ，ウィンドウを選択してください

RRR-7.BAS

a1

a2

RRR-7.BAS:=メインモジュール

[Ver.4.2]

V/表示




C/プログラム選択:

RRR-7.BAS

a1

a2

RRR-7.BAS:=メインプログラム

- ③ 矢印キー  または  でプログラムを選択できます。たとえば「a1」を選んで  するとサブルーチン a1 が表示されます。

RRR-7.BAS:a1

```
SUB a1 (z%, zz%, zzz%)
  zz% = 0
  FOR i = 1 TO z%
    zz% = zz% + i
  NEXT i
  STOP
  CALL a2(z%, yy%)
  zzz% = yy%
END SUB
```

- ④ なお  によっても同様の一覧が選べます。

【例題 43】 サブルーチンの選択

【例題 42】のプログラムを使って，メインルーチン，サブルーチンを順に呼び出せ。

【解 説】 ◎ メインプログラム、サブルーチンを順に呼び出します。

- ① 表示メニューを選びます。
- ② ここで”E/次の SUB”を選びます。
- ③ 現在表示されている次のサブルーチンまたはファンクションが表示されます。たとえば、a1 のサブルーチンが表示されている状態で、この手順を行うとサブルーチン a2 が表示されます。
- ④ なお、**SHIFT** + **f・2** で同じことができます。このくり返しで、メインプログラム、サブルーチン、ファンクションを呼び出します。

7 章 編 集

【例題 44】 編集のとり消し

次のプログラムをつくり、次に⑧のように 1 行目を修正している途中でこれを取り消して、元へもどせ。

```
PRINT "Quick Basic ハ ヘンリナ ケンコデス。"
END
```

⑧ PRINT "Quick C ケンコデス。"
END

【解 説】 ⑧ アンドウ機能で修正を取り消します。

- ① 問題のプログラムをつくります。
- ② 次にカーソルを"B"におき **[DEL]** キーをおして文字を消し、**[C]** をおして⑧のようにします。
- ③ ここで編集メニューを選択し、"U/元に戻す"を選びます。
最初の状態にもどります。

【例題 45】 編集機能によるコピーとペースト

項③のプログラムを使い、END の前に 1 行目から 3 行目までを挿入して、数 x , y を入力して積を求めて表示し、次に再度 x と y を入力して積を求めて表示するプログラムとせよ。

【解 説】 ③ E/編集の C/コピーと P/ペーストを使います。

- ① 1 文字単位の挿入、削除、訂正は **[DEL]**, **[BS]**, **[INS]** キーなどで簡単に行えます。しかし、複数行を扱うときは E/編集機能を使うと簡単です。
- ② 今回は INPUT x , y と $z=x*y$ と PRINT x ; "*" ; y ; "=" ; z の 3 行を END の前へ挿入します。
- ③ 第 1 にどの行を扱うかを指定します。以下のプログラムを使い、END の E にカーソルをおき、**[CTRL]** と **[SHIFT]** と **[E]** キーを同時におします。第 3 行の PRINT 以下が反転表示されます。さらに **[CTRL]** と **[SHIFT]** と **[E]** キーをおすと 1 つ上の第 2 行が反転表示されます。もう一度これを行います。第 1 行も反転表示します。この反転表示部分が挿入、削除の対象です。なお矢印キーをおすと元へもどります。

```
INPUT x, y
z = x * y
PRINT x; "*" ; y; "=" ; z
END
```

E にカーソルをおき、**[CTRL]** と **[SHIFT]** をおしたまま **[E]** キーを 3 回おすと第 1～第 3 行が反転表示される。

- ④ 次に E/編集モードを選びます。
- ⑤ ここでは C/コピーを選びます。画面上は変化はありませんが、指定された第 1～第 3 行がクリップボードと呼ばれるメモリにかき込まれます。
- ⑥ カーソルを END の E の上におきます。編集メニューを選びます。今度は P/ペーストを選びます。
- ⑦ **GRPH** + **BS** キーをおすと元へもどります。
- ⑧ クリップボードの 3 行が挿入されます。

```

INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
END

```

3 行挿入

- ⑨ クリップボードの 3 行は残っていますからいつでもどこへでも挿入することができます。たとえば、サブルーチンの画面を出して、サブルーチンの中の指定行へ挿入することも可能です。

【実行結果】

```

? 2,3
2 * 3 = 6
? 5,9
5 * 9 = 45

```

【例題 46】 編集機能を使った削除

【例題 45】のプログラムの 4 行目の INPUT x, y を削除し、次に新しい 4 行目の z=x*y を z=x/y とし、新しい 5 行目の * を / に変更せよ。このプログラムは x と y を入力して積を求めて表示し、次に x/y を z に求めて表示するプログラムである。

【解 説】 ◎ E/編集機能を使って行単位の削除をします。

- ① E/編集の T/カットと E/削除は行単位で削除します。
- ② まず削除する行を指定します。今回は 4 行目の INPUT x, y を削除しますから、5 行目の先頭の z にカーソルをおき **CTRL** + **SHIFT** + **E** キーをおします。4 行目が反転表示されます。
- ③ 編集メニューを選びます。T/カットまたは E/削除を選びます。4 行目の INPUT x, y が削除されます。T/カットの場合は第 4 行がクリップボードに残ります。E/削除ではクリップボードにも残りません。
- ④ 次のようになります。

```

INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
z = x * y
PRINT x; "*"; y; "="; z
END

```

- ⑤ 4 行目の * を / に、5 行目の * を / に変更します。

```

INPUT x, y
z = x * y
PRINT x; "*"; y; "="; z
z = x / y
PRINT x; "/"; y; "="; z
END

```

【実行結果】

```

8,5          8 と 5 を入力の場合
■ * 5 = 40
8 / 5 = 1.6

```

【例題 47】 サブルーチンへ行単位でコピー

数 x を入力し、 x^2 を表示し、次に、サブルーチン a へ移って、 x^3 を表示するプログラムをつくれ。ただし、サブルーチンでは、メインの INPUT 文に PRINT 文をコピーして 1 部修正して使うものとする。

【解 説】 ㉠ メインプログラムの行をサブルーチンへコピーします。

① 次のようにメインプログラムとサブルーチンをつくります。

```

INPUT x
y = x * x
PRINT x; "^ 2 ="; y
CALL a(x)
END

```

END の後に SUB a(x) をかき  またはカーソルの行をかえると下の画面となります。

```
SUB a (x)
```

```
END SUB
```

② メインプログラムの第 2 行と第 3 行をサブルーチンへ挿入します。したがって、まずメインプログラムの第 3 行目の CALL a(x) の C にカーソルをおき、**CTRL** + **SHIFT** + **E** を 2 回おします。2 行～3 行が反転表示されます。

③ E/編集メニューを選び、C/コピーを選んで、2 行をクリップボードにかき込みます。

④ **SHIFT** + **F2** キーで SUB a(x) のサブルーチンがアクティブ画面となります。

⑤ カーソルをサブルーチンの空白行におき編集メニューから C/コピーを選びます。

⑥ 次のように 2 行が挿入されます。

```

SUB ■ (x)
y = x * x
PRINT x; "^ 2 ="; y

```

```
END SUB
```

⑦ 空白にカーソルをおき **DEL** で空白をとり、次に、第 2 行に $* x$ を追加し、第 3 行の 2 を 3 に変更します。

```

SUB a
y = x * x * x
PRINT x; "^ 3 ="; y
END SUB

```

【実行結果】

```
? 5
5 ^ 2 = 25
5 ^ 3 = 125
```

5 を入力の場合

8 章 検 索

【例題 48】 検 索

次のプログラムを作成し、"i"を検索せよ。ただし、このプログラムは1から10までの各値の和、2乗の和、3乗の和を求めるものです。

```
FOR i = 1 TO 10
  x = x + i
  y = y + i * i
  z = z + i * i * i
NEXT i
PRINT x
PRINT y
PRINT z
END
```

【解 説】 ◎ 文字の検索をします。

- ① 文字、文字列の検索を行います。この例ではプログラムが短いため、簡単に1目でわかりますが、長いプログラムの中で特定の文字や文字列をさがすのに便利です。
- ② 検索メニューを選択します。
- ③ "F/検索" ... を選びます。次の検索文字列指定画面となります。

[Ver.4.5]

F/検索
検索する文字列と検索条件を設定してください

F/検索対象:

	範囲
<input type="checkbox"/> M/大小文字区別	<input type="checkbox"/> 1. 現在のウィンドウ
<input type="checkbox"/> W/全体一致	<input checked="" type="checkbox"/> 2. 現在のモジュール
	<input type="checkbox"/> 3. 全モジュール


< 確認 > < 取消 > < H/ヘルプ >

[Ver.4.2]

F/検索文字列

検索	
<input type="checkbox"/> 1/アクティブウィンドウ	<input type="checkbox"/> M/大小文字区別
<input checked="" type="checkbox"/> 2/カレントウィンドウ	<input type="checkbox"/> W/全体一致
<input type="checkbox"/> 3/全体	

確認 取消

- ④ 今回は"i"で検索をしますので"i"をかき込みます。
- ⑤  によって、1行目の FOR i=1 TO 10 の i にカーソルが移ります。検索メニューで"R/次

を検索”を選ぶと次の i にカーソルが移ります。

- ⑥ なお、“M/大小文字区別”を選択していれば、i は小文字ですから PRINT x の I は検索しません。
 [TAB] キーで“M/大小文字区別”の前の[]にカーソルをおき、スペースをおすと[X]となり、大文字・小文字の区別をします。再度スペースをおすと解除されます。

【例題 49】 文字列の検索

【例題 48】のプログラムを使って、PRINT を検索せよ。

【解 説】 ◎ 文字列を検索します。

- ① 【例題 48】の手順で検索文字列として“PRINT”を指定します。[J] によって最初の PRINT にカーソルが移ります。
 ② 検索メニューを選び、次に“R/次を検索”を選びます。次の“PRINT”にカーソルを移します。以下同様に [F.3] をおすと順に次の“PRINT”を検索します。

【例題 50】 置 換

【例題 48】のプログラムを使って、PRINT を LPRINT に置きかえよ。

【解 説】 ◎ 文字列を検索し、その文字列を指定した文字列に置きかえます。

- ① 検索メニューを選びます。
 ② “C/置換...”を選びます。
 ③ “F/置換前”に“print”[TAB]、T/置換後に“lprint”[TAB] をします。次に [TAB] キーで C/一括置換を選んで [J] または [C] とします。

[Ver.4.5]

C/置換

置換する文字列と置換条件を設定してください

F/置換前: PRINT

T/置換後: LPRINT

[] M/大小文字区別
 [] W/全体一致

範囲

- () 1. 現在のウィンドウ
 (*) 2. 現在のモジュール
 () 3. 全モジュール

< V/逐次確認 > < 一括置換 >

< 取消 > < H/ヘルプ >

[Ver.4.2]

F/置換前 print

T/置換後 lprint

置換

- () 1/アクティブウィンドウ
 (*) 2/カレントウィンドウ
 () 3/全体

- [] M/大小文字区別
 [] W/全体一致

V/逐次確認

C/一括置換

取消



- ④ プログラム中の PRINT はすべて LPRINT に置換されます。

```
FOR i = 1 TO 10
  x = x + i
  y = y + i * i
  z = z + i * i * i
NEXT i
LPRINT x
LPRINT y
LPRINT z
END
```

【例題 51】 部分置換

【例題 50】でつくったプログラムのうち LPRINT y のみ PRINT y にかえよ。

【解 説】 ◎ 検索文字列群の 1 部のみを置換します。

- ① 【例題 50】でつくったプログラムを使います。
 ② 検索メニューを選び、次に“C/置換”を選択します。
 ③ “F/置換前”として“lprint”、“T/置換後”として“print”を指定します。
 ④ 次に V/逐次確認を選んで  または  とします。カーソルが PRINT x におかれて次の画面となります。

[Ver.4.5]

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 O/オプション H/ヘルプ
 Untitled

```
FOR i = 1 TO 10
  x = x + i
  y = y + i * i
  z = z + i * i * i
NEXT i
LPRINT x
LPRINT y
LPRINT z
END
```

C/置換

処理を選択してください

< C/置換 > < S/スキップ > < 取消 > < H/ヘルプ >

[Ver.4.2]

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 H/ヘルプ (f・1)
 RRR-4.BAS

```

FOR i = 1 TO 10
  x = x + i
  y = y + i * i
  z = z + i * i * i
NEXT i
LPRINT x
LPRINT y
LPRINT z
END

```

C/置換する

S/置換しない

取消

- ⑤ LPRINT x は置換しませんので "S/スキップ" (バージョン 4.5), "S/置換しない" (バージョン 4.2) を選んで **[J]** または **[S]** とします。次の LPRINT y へ進みます。
- ⑥ カーソルが "LPRINT y" におかれます。
- ⑦ ここでは LPRINT y を PRINT y に置換しますから "C/置換" (バージョン 4.5), "C/置換する" (バージョン 4.2) にカーソルを移し **[J]** または **[C]** とします。"LPRINT y" が "PRINT y" に置換されます。

```

FOR i = 1 TO 10
  x = x + i
  y = y + i * i
  z = z + i * i * i
NEXT i
LPRINT x
PRINT y
LPRINT z
END

```

【例題 52】 ラベルの検索


次のプログラムで検索文字列 "ex" を指定して、ラベル "ex:" を検索せよ。ただし、このプログラムは 1 から 10 までの和, 2 乗の和, 3 乗の和を求めて表示するものであるが, 3 乗の和が 200 以上となったら結果を表示するものです。

```

FOR i = 1 TO 10
  x = x + i
  y = y + i * i
  z = z + i * i * i
  IF z >= 200 THEN GOTO ex
NEXT i
ex:
PRINT x
PRINT y
PRINT z
END

```


【解 説】 ㊦ ラベルの検索を行います。

- ① 検索メニューを選択します。
- ② 続いて”L/ラベル検索 (バージョン 4.5)”, ”L/ラベル ... ” (バージョン 4.2) を選びます。
- ③ ラベル名として ex: をかき  します。ラベル”ex:”を検索して、カーソルを移します。次のラベル入力画面となります。

[Ver.4.5]

L/ラベル検索

検索する文字列と検索条件を設定してください

F/検索対象: ex:

範囲

- | | |
|-----------------------------------|---|
| <input type="checkbox"/> M/大小文字区別 | <input type="checkbox"/> 1. 現在のウィンドウ |
| <input type="checkbox"/> W/全体一致 | <input checked="" type="checkbox"/> 2. 現在のモジュール |
| | <input type="checkbox"/> 3. 全モジュール |

< 確認 > < 取消 > < H/ヘルプ >

[Ver.4.2]

F/検索文字列: ex:

検索

- ☐ 1/アクティブウィンドウ
☒ 2/カレントウィンドウ
☐ 3/全体

- ☐ M/大小文字区別
☐ W/全体一致

確認

取消

9 章 デバッグ・関数

【例題 53】 トレース

次のプログラムは seisu% に整数を入力し、階乗を求めるプログラムである。これを 1 ステップずつトレースせよ。

```

DECLARE FUNCTION kaijou (s%)
INPUT seisu%
PRINT seisu%; "!="; kaijou(seisu%)
END

FUNCTION kaijou (s%)
  IF s% = 0 THEN
    kaijou = 1
  ELSE
    kaijou = s% * kaijou(s% - 1)
  END IF
END FUNCTION

```

【解 説】 ◎ トレースを行います。

- ① トレースとは 1 ステップずつ実行することです。プログラムの流れを確認できます。【例題 55】で行う、整数のウォッチと合わせて行うことでプログラムのデバッグを簡単に行います。
- ② トレース機能は **f・8** キーと **f・10** キーです。**f・8** キーはサブルーチンやファンクションの中までトレースするのに対し、**f・10** はメインプログラムの中だけトレースします。
- ③ **f・8** キーまたは **f・10** キーをおすたびに実行されているステートメントが緑色になります。また途中で **f・5** キーをおすと通常の実行に移ります。途中でトレースをやめて高速で終了させる場合に使います。
- ④ 実際にトレースを行います。
 まず最初の **f・8** をおします。"INPUT seisu%" が緑色となります。
 次に再び **f・8** をおします。
 ? が表示されます。整数の入力です。今回は 6 **↵** とします。
 "PRINT seisu%; "!="; kaijou(seisu%)" が緑となります。続いて **f・8** をおします。
 FUNCTION kaijou(s%) の中のトレースに入り "IF s%=0 THEN" が緑色となります。
 続いて **f・8** をおします。
 "kaijou=s%*kaijou(s%-1)" が緑色となります。
 以下 **f・8** をおすたびに "IF s%=0 THEN" と "kaijou=s%*kaijou(s%-1)" をくり返し緑色とします。s% が 0 となると次の **f・8** で kaijou=1 が緑色となります。
f・8 をおします。
 "ELSE" が緑色となります。

f.8 をおします。

"END FUNCTION"が緑色となります。

f.8 をおします。

"END IF"が緑色になります。

次からの **f.8** によって"END FUNCTION"と"END IF"をくり返した後ファンクションからぬけ出てメインプログラムへもどります。

"END"が緑色となります。

f.8 をおします。全体が白くなって終了です。

- ⑤ **f.10** を使うと、整数を入力した後、ファンクションは自動実行して、すぐ "PRINT seisu %;"!=";kaijou(seisu%)"が緑色となり、次の **f.10** で"END"が緑色となって、次の **f.10** で終了です。

【例題 54】 低速トレース

【例題 53】のプログラムを使って、低速でトレースを自動実行せよ。

【解 説】 ㊟ トレースを低速で実行します。

- ① トレースを低速で行います。

デバッグメニューを選択します。

- ② T/トレースを選びます。元の画面にもどります。

- ③ **f.5** をおすと"INPUT seisu%"が緑色となり、入力待ちとなりますので整数を入力します。
たとえば 6 **J** です。

- ④ あとは自動的に 1 ステップずつ進めていきます。実行のステートメントが緑色にかわります。

- ⑤ なお、デバッグメニューを選ぶと"T/トレース"の前に*(バージョン 4.5)、>(バージョン 4.2)がついていて、トレースが選ばれていることがわかります。

- ⑥ 解除は T/トレースを選んで **J** します。

【例題 55】 ウォッチ

次のプログラムを 1 ステップずつ実行し、そのつど変数 **i** と **x** の値をチェックせよ。このプログラムは **i** を 1 から 10 までとし、**x** にその和を求めるものです。

```
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

【解 説】 ㊟ ウォッチ機能で変数の値をチェックします。

- ① ウォッチ機能は指定した変数や式の値をチェックするものです。
- ② デバッグメニューを選択します。
- ③ "A/ウォッチ"を選びます。次の画面となります。

[Ver.4.5]

A/ウォッチの追加

ウォッチウィンドウに追加する式を入力してください

< 確認 > < 取消 > < H/ヘルプ >

[Ver.4.2]

ウォッチウィンドウに登録する式・変数の設定:

確認
取消

- ④ ウォッチすなわちチェックをする変数名 **i** をかきます。
- ⑤ J をすると次図のようにウォッチする **i** が左上部に表示されます。

```
RRR-2.BAS i:
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

RRR-2.BAS

- ⑥ 同様の手順、デバッグメニューから A, X J によって、ウォッチをする第2の変数 **x** が左上部に表示されます。

```
RRR-2.BAS i:
RRR-2.BAS x:
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

RRR-2.BAS

- ⑦ 実行します。 f・8 をおします。1ステップずつ進みます。まず変数 **i**, **x** とも0ですから次図のように、**i**, **x** の値として0が表示されます。"FOR i=1 TO 10"が緑色になっています。

```
RRR-2.BAS i: 0
RRR-2.BAS x: 0
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

RRR-2.BAS

- ⑧ 続いて f・8 をおします。 **i** が1となります。


```

RRR-2.BAS i: 1
RRR-2.BAS x: 0
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END

```

- ⑨ **f・8** をおします。 $x=x+i$ により x が 1 となります。

```

RRR-2.BAS i: 1
RRR-2.BAS x: 1
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END

```

- ⑩ **f・8** をおします。 " $x=x+i$ " が緑色となり i が 2 となります。

```

RRR-2.BAS i: 2
RRR-2.BAS x: 1
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END

```

- ⑪ **f・8** をおします。 " $NEXT i$ " が緑色となり x は $1+2$ の結果の 3 となります。

```

RRR-2.BAS i: 2
RRR-2.BAS x: 3
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END

```

- ⑫ このくり返しで i が 10, x が 55 となります。

```

RRR-2.BAS i: 10
RRR-2.BAS x: 55
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END

```

- ⑬ 次に **f・8** をおすと " $PRINT x$ " が緑色となり i が 11, x が 55 となります。

```

RRR-2.BAS i: 11
RRR-2.BAS x: 55
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END

```

- ⑭ **f・8** をおすと " END " が緑色となり、さらに **f・8** をおすと " END " が白となって終了です。

- ⑮ バージョン 4.5 では I/簡易ウォッチが設けられました。ウォッチする式にカーソルをおき、デ

バグメニューから I/簡易ウォッチを選ぶか、または **SHIFT** + **f.9** でウォッチ対象を決められます。

【例題 56】 ウォッチを設定してトレースの自動実行

【例題 55】のプログラムと **i** と **x** をウォッチに設定して自動トレースせよ。

【解 説】 ◎ ウォッチを設けて自動トレースをします。

- ① 【例題 55】と同じ手順で **i** と **x** のウォッチを設定します。
- ② 次に【例題 53】の手順でトレースを設定します。
- ③ **f.5** をおします。 **i** と **x** の値を順に表示しながらトレースを実行していきます。

【例題 57】 ウォッチの削除

【例題 56】のように **i** と **x** をウォッチして自動トレースした後、**i** のウォッチを削除せよ。

【解 説】 ◎ ウォッチの削除をします。

- ① 【例題 56】の手順で **i** と **x** をウォッチに設定して自動トレースします。
- ② デバグメニューを選びます。
- ③ D/ウォッチの削除を選びます。次の画面となり、ウォッチを削除する変数名がリストアップされます。

[Ver.4.5]

RRR-2.BAS

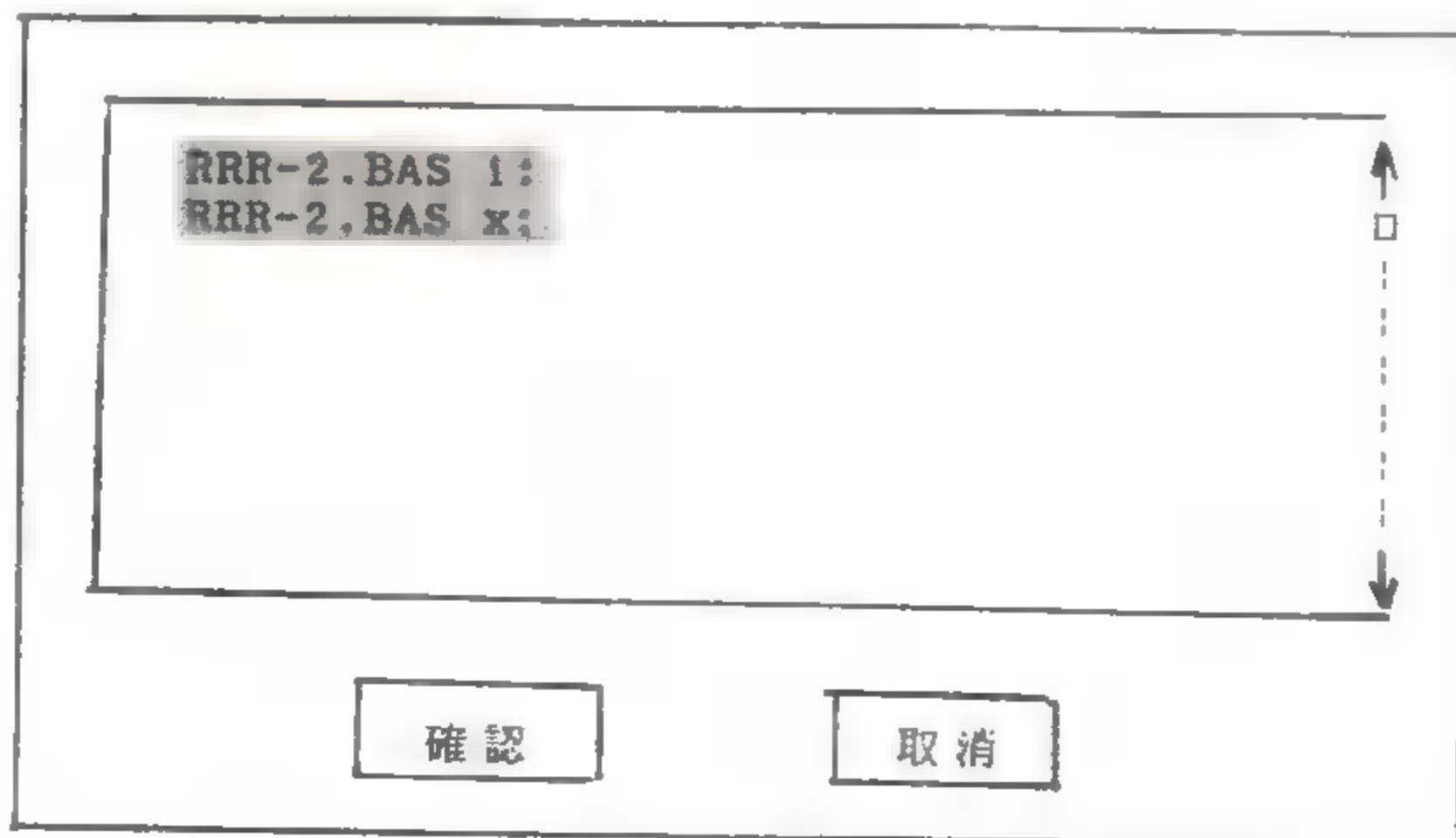
D/ウォッチの削除


削除する式を選択してください

RRR-2.BAS i:
RRR-2.BAS x:

< 確認 > < 取消 > < H/ヘルプ >

[Ver.4.2]



- ④ 矢印キーを使って RRR-2.BAS i: にカーソルを移し  します。i がウォッチから除かれます。

RRR-2.BAS x:

```
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

RRR-2.BAS

【例題 58】 条件式を設定してトレース（ウォッチポイント）

【例題 55】のプログラムを i=5 まで自動トレースで x をウォッチし、次に i=8 までを自動トレースでウォッチせよ。

【解 説】 ◎ 条件式を設定して、真になるまでトレースします。

- ① 条件式を設定して、式が真になるまでトレースします。これをウォッチポイントと呼びます。
- ② 【例題 55】の手順で x をウォッチに設定します。
- ③ 【例題 53】の手順で自動トレースを設定します。
- ④ デバッグメニューを選び、"W/ウォッチポイント"を選びます。次の画面となります。

[Ver.4.5]

W/ウォッチポイント

式を入力してください(式が真になるまで実行します)

< 確認 > < 取消 > < H/ヘルプ >

[Ver.4.2]

F/ファイル E/編集 V/表示 S/検索 R/実行 D/デバッグ C/関数 H/ヘルプ (f・1)
RRR-2.BAS x:


RRR-2.BAS

FOR

NEX 条件式の設定:(式=真になるまで実行します)
PRI
END

 認

取消

- ⑤ 条件は i=5 まで実行しますから i=5 と記入します。
- ⑥  します。条件式が左上に表示されます。現在は i=5 ではありませんから <FALSE> (偽) と表示されています。

RRR-2.BAS x:

RRR-2.BAS i = 5: <FALSE>


RRR-2.BAS

```
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

- ⑦ これで準備は完了しましたので **f・8**、または **f・10** または **f・5** で実行します。今回は **f・5** をおします。

トレースが行われ、 $i=5$ となると画面左上の **FALSE** が **TRUE** にかわり i が5になったことを示します。このときの x の値は10です。また $i=5$ のときの $x=x+i$ は行われていません。

```
RRR-2.BAS x: 10
RRR-2.BAS i = 5: <TRUE>
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

- ⑧ 続いて $i=8$ まで実行するものとします。デバッグメニューを選び、**W/ウォッチポイント** を選びます。 $i=8$  とします。

```
RRR-2.BAS x: 10
RRR-2.BAS i = 5: <TRUE>
RRR-2.BAS i = 8: <FALSE>
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

 をしたところ

- ⑨ **f・5** でトレースを再開します。 $i=8$ となると $i=8$ が **TRUE** となり停止します。 x の値は28です。 $i=8$ のときの $x=x+i$ はまだ実行されていません。

```
RRR-2.BAS x: 28
RRR-2.BAS i = 5: <FALSE>
RRR-2.BAS i = 8: <TRUE>
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

- ⑩ **f・5** を再度おします。実行が継続され x は55、 $i=11$ となって、終了します。ただし、 i の値はウォッチしていませんから表示されません。

```
RRR-2.BAS x: 55
RRR-2.BAS i = 5: <FALSE>
RRR-2.BAS i = 8: <FALSE>
RRR-2.BAS
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

【例題 59】 ウォッチの削除

【例題 58】のウォッチを全部削除せよ。

【解 説】 ◎ ウォッチを全部削除します。

- ① 【例題 55】の手順でウォッチを設け、自動トレースをします。
- ② デバッグメニューを選びます。

- ③ "L/全ウォッチ削除"を選びます。全ウォッチが削除されます。

【例題 60】 ブレークポイント

【例題 55】のプログラムを x をウォッチして実行せよ。ただし、 $x=x+i$ のステートメントにブレークポイントをおき、そのステートメントを実行するとき停止するものとする。

【解 説】 ㊟ ブレークポイントを設けてプログラムを 1 度停止します。

- ① ブレークポイントはプログラム中のステートメント行を指定して、その行を実行するとき一度プログラムを停止させます。変数をウォッチに設定すればプログラムが停止したところでの変数の値をチェックできます。
- ② 【例題 55】の手順で変数 x をウォッチに設定します。
- ③ カーソルを $x=x+i$ の行へおきます。
- ④ ここで **f・9** キーをおします。この行が赤くなります。これはブレークポイントが設定されたことを示します。これはデバッグメニューから "B/ブレークポイント" を選んでも同じです。
ブレークポイントはカーソルのある行に設けます。ブレークポイントは複数の行に設けることができますから、カーソルを移動して **f・9** をおすか、デバッグメニューから指定します。
- ⑤ **f・5** で実行します。 $x=x+i$ の行へくると一度停止します。その時の x の値をチェックすることができます。継続は **f・5** をおします。

```
RRR-2.BAS x: 55
FOR i = 1 TO 10
  x = x + i
NEXT i
PRINT x
END
```

RRR-2.BAS

最後は x の値が 55 となって終了します。

【例題 61】 ブレークポイントの解除

【例題 60】でつくったブレークポイントをすべて解除せよ。

【解 説】 ㊟ ブレークポイントをすべて解除します。

- ① 【例題 60】のブレークポイントを利用します。この例ではブレークポイントは 1 つですが複数個あっても同じです。
- ② デバッグメニューを選びます。"C/全ブレークポイントの削除" (バージョン 4.5), "C/全ブレークポイントの解放" (バージョン 4.2) を選びます。すべてのブレークポイントが解放されます。
- ③ 1 行の場合は、その行へカーソルをおき **f・9** をおします。

【例題 62】 ヒストリ

次のプログラムを実行し、終了したら、1 ステートメントずつ逆にさかのぼってプログラムの流れをチェックせよ。なお、このプログラムは乱数を 10 個つくって、0.3 以下のとき x1、0.3 をこえ 0.6 以下のとき x2、それ以外は x3 にその出現回数を数えるものです。

```

FOR i = 1 TO 10
  x = RND(5)
  IF x <= .3 THEN
    x1 = x1 + 1
  ELSE
    IF x <= .6 THEN
      x2 = x2 + 1
    ELSE
      x3 = x3 + 1
    END IF
  END IF
NEXT i
PRINT x1
PRINT x2
PRINT x3
END

```

【解 説】 ◎ ヒストリは最大 20 までのステップの流れを記憶します。

- ① ヒストリは、最新の 20 ステップを記憶し、逆にさかのぼってステップを確認したり、さかのぼってから、また前へ進めることができます。

エラーがおきてプログラムが止まったとき、直前のステップを確認するのに大変便利です。

- ② デバッグメニューを選びます。

- ③ "H/ヒストリ"を選びます。元の画面にもどります。デバッグメニューを選ぶと"H/ヒストリ"の前に"*"(バージョン 4.5)、">"(バージョン 4.2)がついていて、ヒストリが選択されていることを示します。再度"H/ヒストリ"を選ぶとヒストリ機能は解除されます。

- ④ **f・5** でプログラムを実行します。プログラムは実行されて、"END"にカーソルがきます。ここで **SHIFT** キーと **f・8** キーをおすとプログラムのステップを逆にさかのぼっていきます。カーソルが"PRINT x3"→"PRINT x2"→"PRINT x1"…とさかのぼります。途中で **SHIFT** キーと **f・10** キーをおすと、元の流れでプログラムを進めていきます。

【例題 63】 次のステートメント

【例題 62】のプログラムを使って"x3=x3+1"にブレークポイントをおき実行し、次にカーソルを PRINT x1 において次のステートメントを設定してプログラムを継続させよ。

【解 説】 ◎ 次のステートメントを設定します。

- ① 次のステートメントは GOTO 文のようなもので、途中の実行を省略して、カーソルのあるステートメントへ実行を移します。
- ② カーソルを"x3=x3+1"におき **f・9** キーでブレークポイントをおきます。
- ③ **f・5** で実行します。x3=x3+1 の実行で止まります。x3=x3+1 はまだ実行していません。

- ④ カーソルを"PRINT x1"におきます。
- ⑤ デバッグメニューを選び、"S/次のステートメントの設定" (バージョン 4.5), "S/カレントステートメントの設定" (バージョン 4.2) を選びます。次のステートメントが"PRINT x1"となります。
- ⑥ **f.5** をおします。x3=x3+1 を実行して、PRINT x1 に分岐し、PRINT x2, PRINT x3 を実行して終了します。結果は x3 のみ 1 です。

【結 果】

0	x 1 の値
0	x 2 の値
1	x 3 の値

【例題 64】 カーソルのある行まで実行

【例題 62】のプログラムを使い、ウォッチに x1, x2, x3 を指定して、カーソルを x2=x2+1 におき、x2=x2+1 にプログラムの流れがくるまで実行することをくり返して、プログラムを実行せよ。

【解 説】 ◎ **f.7** キーはカーソルのある行までプログラムを実行します。

- ① 【例題 62】の要領で x1, x2, x3 をウォッチの対象とします。またカーソルを"x2=x2+1"の先頭の x におきます。次図となります。

```

RRR-3.BAS x1:
RRR-3.BAS x2:
RRR-3.BAS x3:
RRR-3.BAS
FOR i = 1 TO 10
  x = RND(5)
  IF x <= .3 THEN
    x1 = x1 + 1
  ELSE
    IF x <= .6 THEN
      x2 = x2 + 1
    ELSE
      x3 = x3 + 1
    END IF
  END IF
NEXT i
PRINT x1
PRINT x2
PRINT x3

```

- ② **f.7** キーをおします。プログラムの流れが x2=x2+1 の実行にきたところで止まります。x2=x2+1 はまだ実行されていません。最初の乱数が 0.6 をこえているため x3=x3+1 を実行しているので変数 x3 のみ 1 となっています。

```

RRR-3.BAS x1: 0
RRR-3.BAS x2: 0
RRR-3.BAS x3: 1
RRR-3.BAS
FOR i = 1 TO 10
  x = RND(5)
  IF x <= .3 THEN
    x1 = x1 + 1
  ELSE
    IF x <= .6 THEN
      x2 = x2 + 1
    ELSE
      x3 = x3 + 1
    END IF
  END IF
NEXT i
PRINT x1
PRINT x2
PRINT x3

```

```

        ELSE
            x3 = x3 + 1
        END IF
    END IF
NEXT i
PRINT x1
PRINT x2
PRINT x3

```

- ③ 続いて **f.7** をおします。再び $x2=x2+1$ の実行にきてプログラムが止まります。2 つ目の乱数は 0.3 をこえ 0.6 以下のため $x2$ も 1 となっています。

```

RRR-3.BAS x1: 0
RRR-3.BAS x2: 1
RRR-3.BAS x3: 1
RRR-3.BAS

FOR i = 1 TO 10
    x = RND(5)
    IF x <= .3 THEN
        x1 = x1 + 1
    ELSE
        IF x <= .6 THEN
            x2 = x2 + 1
        ELSE
            x3 = x3 + 1
        END IF
    END IF
NEXT i
PRINT x1
PRINT x2
PRINT x3

```

- ④ 再び **f.7** をおします。 $x2=x2+1$ の実行にきて止まります。3 回目は乱数は 0.3 をこえ 0.6 以下のため $x2$ は 2、4 回目は 0.3 以下のため $x1$ は 1 となっています。5 回目の値が 0.3 をこえ 0.6 以下のため $x2=x2+1$ のステートメントで止まっています。

```

RRR-3.BAS x1: 1
RRR-3.BAS x2: 2
RRR-3.BAS x3: 1
RRR-3.BAS

FOR i = 1 TO 10
    x = RND(5)
    IF x <= .3 THEN
        x1 = x1 + 1
    ELSE
        IF x <= .6 THEN
            x2 = x2 + 1
        ELSE
            x3 = x3 + 1
        END IF
    END IF
NEXT i
PRINT x1
PRINT x2
PRINT x3

```

- ⑤ 再度 **f.7** をおします。 $x1$ は 2、 $x2$ は 3、 $x3$ は 5 となって終了します。すなわち、3 回目に $x2=x2+1$ が実行されると再びこのステートメントを通ることはありませんので、プログラムの終了まで進みます。


```

RRR-3.BAS x1: 2
RRR-3.BAS x2: 3
RRR-3.BAS x3: 5

```

RRR-3.BAS

```

FOR i = 1 TO 10
  x = RND(5)
  IF x <= .3 THEN
    x1 = x1 + 1
  ELSE
    IF x <= .6 THEN
      x2 = x2 + 1
    ELSE
      x3 = x3 + 1
    END IF
  END IF
NEXT i
PRINT x1
PRINT x2
PRINT x3

```

【例題 65】 関 数

次のプログラムを実行し、STOP のつど、呼び出し関係を確認せよ。プログラムは数 $x\%$ を入力し、サブルーチン a1 を呼んで 1 から $x\%$ までの和を求め、メインルーチンにもどり、次にファンクション a2 を呼んで 1 から $x\%$ までの階乗を求めるものである。プログラム名は RRR-6.BAS とする。

```

DECLARE SUB a1 (z AS INTEGER, zz AS INTEGER)
DECLARE FUNCTION a2 (z AS INTEGER)
INPUT x%
STOP
CALL a1(x%, xx%)
PRINT xx%
PRINT a2(x%)
END

```

```

SUB a1 (z%, zz%)
  zz% = 0
  FOR i = 1 TO z%
    zz% = zz% + i
  NEXT i
  STOP
END SUB

FUNCTION a2 (z%)
  y = 1
  FOR i = z% TO 1 STEP -1
    y = y * i
  NEXT i
  a2 = y
  STOP
END FUNCTION

```

```

? 5          5 を入力の例
15
120

```

【解 説】 ◎ 関数メニューで関数の呼び出し関係をチェックします。

- ① **f・5** でプログラムを実行します。 $x\%$ に値を入力した後、4 行目の STOP でプログラムが中断します。次図となり、現在 RRR-6.BAS のプログラムを実行していることがわかります。

C/関数

M/RRR-6.BAS

- ② 続いて **f・5** をおし、プログラムを続行します。サブルーチン a1 の STOP で中断します。ここで関数メニューを選びます。下図となります。

RRR-6.BAS から呼び出された a1 を実行していることがわかります。

C/関数

a1
M/RRR-6.BAS

- ③ **f・5** でプログラムを続行します。次はファンクション a2 の STOP で中断します。ここで関数メニューを選びます。下図となり、RRR-6.BAS から呼び出された a2 を実行中であることがわかります。

C/関数

a2
M/RRR-6.BAS

【例題 66】 関 数

下のプログラムを実行し、サブルーチンの呼び出し関係を確認せよ。ただし、このプログラムはメインルーチンで x% を入力し、サブルーチン a1 を呼び出して 1 から x% までの和を求め、次にサブルーチン a2 を呼び出して、x% の階乗を求めるものである。プログラム名は RRR-7.BAS とする。

```

DECLARE SUB a1 (z%, zz%, zzz%)
DECLARE SUB a2 (z AS INTEGER, zz AS INTEGER)
INPUT x%
STOP
CALL a1(x%, xx%, xxx%)
PRINT xx%
PRINT xxx%
END

SUB a1 (z%, zz%, zzz%)
zz% = 0
FOR i = 1 TO z%
    zz% = zz% + i
NEXT i
STOP
CALL a2(z%, yy%)
zzz% = yy%
END SUB

SUB a2 (y%, yy%)
z% = 1
FOR i = y% TO 1 STEP -1
    z% = z% * i
NEXT i
yy% = z%
STOP
END SUB

```

【解 説】 ◎ 関数メニューでサブルーチンの呼び出し関係を確認します。

- ① **f-5** でプログラムを実行します。メインルーチンの 4 行目の STOP でプログラムが中断します。ここで関数メニューを選びます。下図となり、プログラム RRR-7.BAS が実行中であることがわかります。

C/関数

M/RRR-7.BAS

- ② 次に **f-5** をおして、プログラムを続行します。次にサブルーチン a1 の STOP で中断します。ここで関数メニューを選びます。次の画面となります。RRR-7.BAS から呼び出された a1 を実行中であることがわかります。

C/関数

a1
M/RRR-7.BAS

- ③ **f-5** でプログラムを続行します。サブルーチン a2 の中の STOP で中断します。ここで関数メニューを選びます。次の画面となります。a1 から呼び出された a2 を実行中であることがわかります。

C/関数

a2
a1
M/RRR-7.BAS

II.....プログラム編



1 章 変数の型・四則演算・累乗・整数除算・剰余

%, !, \$, &, #, +, -, *, /, ^, MOD, DEF INT/SNG/LNG/DBL

【例題 1】 変数の型

整数、長整数、単精度浮動小数点数、倍精度浮動小数点数、文字を 1 つずつ入力して表示するプログラムをつくれ。

【解 説】 ◎ 変数の型を学びます。

- ① Quick BASIC では C 言語や FORTRAN のように変数の型を明確にします。
- ② 変数名に%がつくと整数型で-32,768~+32,767 の値を扱います。
- ③ 変数名に&がつくと長整数型で-2,147,483,648~+2,147,483,647 の値を扱います。
- ④ 変数名に!がつくと単精度浮動小数点で-3.402823E+38 から-1.175494E-38 と+1.175494E-38 から+3.402823E+38 までの値を扱います。7 桁が有効です。
- ⑤ 変数名に#がつくと倍精度浮動小数点で-1.797693134862316D+308 から-2.225073858507201D-308 までと+2.225073858507201D-308 から-1.797693134862316D+308 の値を扱います。15 または 16 桁が有効です。
- ⑥ 変数名に\$がつくと文字型で文字列を扱います。最大 32,767 文字まで代入できます。
- ⑦ 変数名が同じでも型宣言文字が異なると別変数として扱われます。たとえば、a%とa#は別の変数として使えます。
- ⑧ a%=156 によって変数 a%に 156 が代入されます。a%=156.3 のように異なる型を代入すると a%には%で決められた型すなわちこの場合は整数として 156 を代入します。
- ⑨ 数値変数に文字を代入することはできません。また、逆に文字変数に数値を代入できません。ただし、数値を" "で囲んで代入すればそれは数値ではなく文字として代入されます。

正しい例

a%=156

a%=ABC

誤

a%="ABC"

a%=ABC.....ABC が変数ならその値を a%に代入する
ABC が文字なら誤り

a\$="156"

a\$=156……156 が数値なら a\$=156 は誤り

a\$="156" とすると 156(いちごろく)の
数字が a\$ に代入される。

⑩ PRINT 変数名で変数の値が表示されます。; は改行をせずにそのまま続けて表示されます。文の最後に ; がいないときは改行されます。

⑪ LPRINT では印刷されます。

【プログラム例】

```
a% = 156
a& = 123456789
a! = 1213.456
a# = 123456.78945#
a$ = "Quick Basic"
PRINT a%; a&; a!; a#; a$
END
```

a%, a&, a!, a#, a\$ に値を代入

表示

【結 果】

156 123456789 1213.456 123456.78945 Quick Basic

【例題 2】 長整数型変数の和

長整数型変数 a& を 1,234,567 とし、b& を 988,777,666 とし、和を求めて表示するプログラムをつくれ。

【解 説】 ◎ 和は+で表わします。

- ① 和は+です。PRINT 文の中で a&+b& のように表わすことができます。a& と b& の和が表示されます。
- ② c&=a&+b& のように結果を左辺の変数に代入し、c& を表示することもできます。
- ③ PRINT 文の中の" "の中の文字はそのまま表示されます。
- ④ ; は改行をせずに続けて表示されます。

【プログラム例】

```
a& = 1234567
b& = 988777666
PRINT a& + b&
END
```

値の代入

和を表示

【結 果】

1234567 + 988777666 = 990012233

【例題 3】 倍精度浮動小数点数の積

52.32145698 と 1.4585424 の積を求めるプログラムをつくれ。

【解 説】 ◎ 倍精度浮動小数点数の積を求めます。

- ① 52.32145698 を a#, 1.4585424 を b# に代入します。積は * で示します。下の式で a# と b# の積を c# に代入します。

c# = a# * b#

- ② 四則演算は +, -, *, / です。

【プログラム例】

```

a# = 52.32145698# 値を a# と b# に代入
b# = 1.4585424#
c# = a# * b# a# と b# を c# に代入
PRINT a#; "*"; b#; " = "; c# 結果の表示
END

```

【結果】

52.32145698 * 1.4585424 = 76.31306343510596

【例題 4】 累 乗

3 の 8 乗の値を求めるプログラムをつくれ。

【解 説】 ◎ 累乗は ^ で求めます。

- ① x% は整数型変数です。x% を 3 とします。
- ② b% = x% ^ 8 では x% の値の 8 乗の値を b% に代入します。^ は累乗を示します。この例では 8 乗です。5.3 乗のように浮動小数点数も使えます。たとえば、b! = a! ^ c! です。
- ③ b& = x% ^ 8 も使えます。x% または 8 の部分が大きくなると b& の方がオーバーフローしない場合が多くなります。

【プログラム例】

```

x% = 3
b% = x% ^ 8 3^8 を b% へ代入
PRINT x%; "^ 8="; b% 表示
END

```

【結果】

3 ^ 8 = 6561

【例題 5】 整数除算と剰余 ¥, MOD

98765589 ÷ 5476555 の商と剰余を求めるプログラムをつくれ。

【解 説】 ◎ 整数除算と剰余を求めます。

- ① c& = a& ¥ b& は長整数 a& ÷ b& の商を c& へ代入します。
- ② ¥ は整数除算です。16 ¥ 3 は 5 です。16.7 ¥ 3.2 は 17 ¥ 3 と同じで 5 です。小数点以下は四捨五入されてから計算されます。
- ③ d& = a& MOD b& では a& ÷ b& の剰余を d& へ代入します。MOD は整数除算の剰余です。

【プログラム例】

```

a& = 98765589
b& = 5476555
c& = a& ¥ b&
d& = a& MOD b&
PRINT a& ; "/" ; b& ; " = " ; c& ; "アマリ" ; d&
END

```

a&, b& に値を代入
a& ¥ b& を求めて c& へ代入
a& ¥ b& の剰余を d& へ代入
結果の表示

【結果】

98765589 / 5476555 = 18 アマリ 187599

【例題 6】 型の異なる変数型へ結果を代入

589 を a%, 246 を b% として a%/b% を求めて c# に代入して、表示するプログラムをつくれ。

【解説】 ◎ 整数型どうしの割り算の結果を倍精度浮動小数点型変数に代入します。

- ① a% を 589, b% を 246 として a%/b% を求めます。このとき計算結果を c% に代入すると c% は 2 です。c# に代入すると倍精度浮動小数点型の値 2.394309043884277 が代入されます。計算結果は代入される変数の型によって決まります。

		a%=589, b%=246 としたときの値
整数型	c%=a%/b%	2
長整数	c&=a%/b%	2
単精度浮動小数点数	c!=a%/b%	2.394309
倍精度浮動小数点数	c#=a%/b%	2.394309043884277

【プログラム例】

```

a% = 589
b% = 246
c# = a% / b%
PRINT a% ; "/" ; b% ; " = " ; c#
END

```

a%, b% に値を代入
a%/b% を c# に代入
結果の表示

【結果】

589 / 246 = 2.394309043884277

【例題 7】 変数の型宣言

変数 A を整数型、変数 B を長整数型、変数 C を単精度浮動小数点数型、変数 D を倍精度浮動小数点数型、変数 E を文字型と宣言して、a~e を下の値として表示するプログラムをつくれ。

A 168, B 2583296, C 15.389, D 123456789
E アイウエオ

【解説】 ◎ 変数の型宣言をします。

- ① DEFINT A は変数 A (大文字小文字の区別はないから a も同じ) を INT 型、整数型と宣言しま

す。実際には変数 A のみでなく A で始まるすべての変数を整数型とします。したがって、今後 a%とせず a のみで整数型を示します。

- ② DEFNG B は同様に変数 B を LNG 型、長整数型とします。
- ③ DEFSNG C は同様に変数 C を SNG 型、単精度浮動小数点数型とします。
- ④ DEFDBL D は同様に変数 D を DBL 型、倍精度浮動小数点数型とします。
- ⑤ DEFSTR E は同様に変数 E を STR 型、文字型とします。

【プログラム例】

```
DEFINT A
DEFLNG B
DEFSNG C
DEFDBL D
DEFSTR E
a = 168
b = 2583296
c = 15.389
d = 123456.789#
e = "アイウエオ"
PRINT a; b; c; d; e
END
```

A～E の型宣言

a～e に値の代入

表示

【結果】

168 2583296 15.389 123456.789 アイウエオ

【例題 8】 文字型を範囲で指定と文字変数の和

A, B, C で始まる変数を文字型と宣言し, alanguage を "Microsoft", blanguage を "Quick c", clanguage を "Quick BASIC" として alanguage+blanguage, alanguage+clanguage を表示するプログラムをつくれ。

【解説】 ◎ A から C までの文字で始まる変数名を文字型とします。また、文字変数の和を求めます。

- ① DEFSTR A-C は A から C まですなわち、A, B, C で始まる変数名を文字型と定義します。
- ② したがって、alanguage, blanguage, clanguage は各々 a, b, c で始まっていますから文字型変数です。
- ③ alanguage+blanguage は文字変数の和で各々の文字列をつなぎます。

【プログラム例】

```
DEFSTR A-C
alanguage = "Microsoft"
blanguage = "Quick C"
clanguage = "Quick Basic"
PRINT alanguage + blanguage
PRINT alanguage; clanguage
END
```

A～C を文字型で宣言

文字列を代入

表示 和は文字列をつなぐ

【結 果】

```
Microsoft Quick C
Microsoft Quick Basic
```

【例題 9】 配列の型宣言

配列 a(10) を文字列配列として、a(0)～a(9) に下のデータを代入して、次に "T" で始まる文字列を表示するプログラムをつくれ。

データ：Rome, Washington, Tokyo, London, Hongkong,
Chicago, Taipei, Paris, Bon, Moscow

【解 説】 ◎ 配列の型宣言をします。

- ① 文字型配列は次のように宣言します。配列 a(0)～a(9) が文字型であると宣言するとともに、各文字列の長さは 10 文字となります。配列についてはⅡ編 8 章で扱います。

DIM	a(10)	AS STRING	* 10
配列宣言	配列名 a\$(10)と \$をつける 必要はない	文字列として 宣言	10 文字の 文字列として固定

- ② a(0)～a(9) に文字列を代入します。FOR 文によるくり返しはⅡ編 5 章で扱います。a(0) は Rome, a(1) は Washington ですが、各文字列は 10 文字ですから a(0) は "Rome" となります (□ は空白)。a(1) は "Washington" です。
- ③ ASC (文字) は文字のアスキーコードを示す値です。たとえば、"A", "America" は先頭文字 "A" のアスキーコード 65 です。

【プログラム例】

```
DIM a(10) AS STRING * 10  ————— 各 10 文字の文字列の文字列配列 a(9) を宣言
FOR i = 0 TO 9 ————— a(0)～a(9) にデータを読み込む
    READ a(i)
NEXT i
FOR i = 0 TO 9 ————— "T" で始まる文字列を表示
    IF ASC(a(i)) = 84 THEN PRINT a(i)
NEXT i
END
DATA Rome, Washington, Tokyo, London, Hongkong
DATA Chicago, Taipei, Paris, Bon, Moscow
```

【結 果】

```
Tokyo
Taipei
```

【例題 10】 配列を整数型で宣言

配列 a(10) を整数型で宣言し、下のデータを読み次に、その値をアスキーコードとする文字を順に表示するプログラムをつくれ。

データ：87, 65, 83, 72, 73, 78, 71, 84, 79, 78

【解 説】 ◎ 配列を整数型で宣言します。

- ① 配列 a(10) を整数型で宣言するときは次のようにします。また、その他の型の場合も INTEGER の部分を変えるだけです。文字型については【例題 9】で扱っています。配列については II 編 8 章で扱います。

```
DIM a(10) AS 

|         |       |               |
|---------|-------|---------------|
| INTEGER | ..... | 整数型で宣言        |
| LONG    | ..... | 長整数型で宣言       |
| SINGLE  | ..... | 単精度浮動小数点数型で宣言 |
| DOUBLE  | ..... | 倍精度浮動小数点数型で宣言 |


```

- ② CHR\$(値) はその値をアスキーコードとする文字を持ちます。たとえば、CHR\$(65) は A です。

【プログラム例】

```
DIM a(10) AS INTEGER  ----- 配列 a(9) を整数型で宣言
FOR i = 0 TO 9 ----- データの読み込み
    READ a(i)
NEXT i -----
FOR i = 0 TO 9 ----- 表示
    PRINT CHR$(a(i));
NEXT i -----
END
DATA 87, 65, 83, 72, 73, 78, 71, 84, 79, 78
```

【結 果】

WASHINGTON

〔演習問題〕

- (1) 長整数 x& を 123456789, y& を 9874563 として x&-y& を b& に代入し、結果を表示するプログラムをつくれ。
- (2) a& を 85236987, b& を 654865 として a&/b& を c& に求め結果を表示するプログラムをつくれ。
- (3) (2) と同じ式の値を c# に求めて表示するプログラムをつくれ。
- (4) a% を 1234, a! を 123.456, a& を 123456789, a# を 123.45678912, a\$ を "Quick Basic" として、各々を表示するプログラムをつくれ。
- (5) A と U で始まる変数を倍精度浮動小数点数型, R から S まですで始まる変数名を単精度浮動小数点数型と宣言して, uriage1 を 1986500, uriage2 を 856200, uriage3 を 752500,

arari1 を 350200, arari2 を 97600, arari3 を 102500 として, ritsul に uriage1 に対する arari1 の率, seihin1 に uriage2 に対する arari2 の率, seihin2 に uriage3 に対する arari3 の率を小数点第 1 位まで切りすてで求めるプログラムをつくれ.

- (6) $25638.9 \div 1965.3$ の整数除算をするプログラムをつくれ.
- (7) $3892 \div 19$ の商と剰余を求めるプログラムをつくれ.
- (8) 3^9 を求めるプログラムをつくれ.
- (9) 整数 $x!$ を 3.5, $y!$ を 2.5 として $x!^y!$ を求めるプログラムをつくれ.
- (10) $186.3292654 \times 5\%$ を求め結果を単精度実数で表示するプログラムをつくれ.
- (11) 配列 a(10) を単精度実数型で宣言し, 次の数値データを 10 個読み込んで表示するプログラムをつくれ.

データ: 87, 65, 83, 72, 73, 78, 71, 84, 79, 78

- (12) A, B, C で始まる変数名を文字型と宣言し, America, Austria, Britain, Burma, Canada, China に各々次の文字列を代入して表示するプログラムをつくれ.
- USA, Europe, England, Asia, North America,
Mainland China
- (13) x, y, z で始まる変数を整数型変数と宣言し, $a! = 38.29$, $b! = 11.33$, $c! = 8.76$ として $a! * b!$ の値を x に, $a! \div c!$ の値を y に $b! * c!$ の値を z に代入して表示するプログラムをつくれ.

〔解 答〕

(1) 【プログラム例】

```
x& = 123456789
y& = 9874563
b& = x& - y&
PRINT x& " - " y& " = " b&
END
```

値の代入
差
結果の表示

【結 果】

123456789 - 9874563 = 113582226

(2) 【プログラム例】

```
a& = 85236987
b& = 654865
c& = a& / b&
PRINT a& " / " b& " = " c&
END
```

a& と b& に値を代入
a&/b& を c& に代入
結果の表示

【結 果】

85236987 / 654865 = 130

(3) 【プログラム例】

```

a& = 85236987 ————— a& と b& に値を代入
b& = 654865 —————
c# = a& / b& ————— a&/b& を c# に代入
PRINT a&; "/" ; b&; "=" ; c# ——— 結果の表示
END

```

【結 果】

```
85236987 / 654865 = 130.1596313744054
```

(4) 【プログラム例】

```

a% = 1234 ————— 整数型変数に値を代入
a! = 123.456 ————— 単精度型実数に値を代入
a& = 1213456789 ————— 長整数型変数に値を代入
a# = 123.45678912# ————— 倍精度型実数に値を代入
a$ = "Quick Basic" ————— 文字型変数に値を代入
PRINT a%; a! ————— 表示
PRINT a&; a#
PRINT a$
END

```

【結 果】

```

1234 123.456
1213456789 123.45678912
Quick Basic

```

(5) 【プログラム例】

```

DEFDBL A, U ————— A, U を倍精度浮動小数点数型として宣言
DEFSNG R-S ————— R, S を単精度実数として宣言
uriage1 = 1986500 ————— 初期値の代入
uriage2 = 856200
uriage3 = 752500
arari1 = 350200
arari2 = 97600
arari3 = 102500
ritsu1 = INT(arari1 / uriage1 * 1000) / 10 ————— 率の計算
seihin1 = INT(arari2 / uriage2 * 1000) / 10
seihin2 = INT(arari3 / uriage3 * 1000) / 10
PRINT ritsu1; "% "; seihin1; "% "; seihin2; "% " ——— 結果の表示
END

```

【結 果】

```
17.6 % 11.3 % 13.6 %
```

(6) 【プログラム例】

```

PRINT 25638.9 ¥ 1965.3 ——— 整数除算
END

```

【結 果】

```
13
```

(7) 【プログラム例】

```

a% = 3892
b% = 19
x1% = a% / b% —— 商
x2% = a% MOD b% —— 剰余
PRINT x1%, x2%
END

```

【結 果】

204

16

(8) 【プログラム例】

```

a% = 3
b% = a% ^ 9 —— 39 を b% に代入
PRINT b%
END

```

【結 果】

19683

(9) 【プログラム例】

```

x! = 3.5 —— 値の代入
y! = 2.5 ——
PRINT x! ^ y! —— 結果の表示
END

```

【結 果】

22.91765

(10) 【プログラム例】

```

x# = 186.3292654#
x% = 5
x! = x# * x% —— x# * x% を x! に代入
PRINT x!
END

```

【結 果】

931.6463

(11) 【プログラム例】

```

DIM a(10) AS SINGLE —— 配列の宣言
FOR i = 0 TO 9 —— データの読み込み
    READ a(i)
NEXT i
FOR i = 0 TO 9
    PRINT a(i); —— 表示
NEXT i
END
DATA 87, 65, 83, 72, 73, 78, 71, 84, 79, 78

```

【結 果】

87 65 83 72 73 78 71 84 79 78

(12) 【プログラム例】

```

DEFSTR A-C ———— 型宣言
america = "USA" ———— 値の代入
austria = "Europe"
britain = "England"
burma = "Asia"
canada = "North America"
china = "Mainland China"
PRINT america, austria ———— 表示
PRINT britain, burma
PRINT canada, china
END
    
```

【結 果】

```

USA           Europe
England       Asia
North America Mainland China
    
```

(13) 【プログラム例】

```

DEFINT X-Z ———— x, y, z で始まる変数名を整数型に宣言
a! = 38.29 ———— 値の代入
b! = 11.33
c! = 8.76
x = a! * b! ———— 積, 商, 積を x, y, z に代入
y = a! / c!
z = b! * c!
PRINT x, y, z
END
    
```

【結 果】

```

434           4           99
    
```

2 章 定 数

CONST

【例題 11】 記号定数

Pi を定数 3.14159 として、半径 30 の円の面積と円周長を求めるプログラムをつくれ。

【解 説】 ◎ 記号定数を使います。

- ① 定数に記号をつけ、常にその値を保ちます。
- ② `CONST Pi!=3.14159` とすると Pi の値は常に 3.14159 に固定されます。途中で `Pi=5.2` のように値を入れかえることはできません。つまりうっかり値を変えてしまうといったミスはおこりません。変数を使うとそういうことがおこることがあります。また、3.14159 のように値の桁が多い場合は記号定数に代人しておいた方が使いやすくなります。
- ③ `CONST Pi!=3.14159` の “!” によって Pi の型を単精度実数型とします。
- ④ `CONST Pi!=3.14159` では Pi が定数の名前で、! は定数のデータ型を示す型宣言文字です。
- ⑤ 記号定数で使われた名前、本例では Pi は以降、変数として使用できません。Pi%, Pi!, Pi#, Pi\$ のいずれも使えません。

【プログラム例】

```
CONST pi! = 3.14159 ——定数 3.14159 を pi! に固定する
s! = 30 * 30 * pi ——面積の計算
l! = 2 * 30 * pi ——円周長の計算
PRINT s! ——結果の表示
PRINT l!
END
```

【結 果】

```
2827.431
188.4954
```

【例題 12】 整数型の記号定数

九九の表をつくるプログラムをつくれ。

【解 説】 ◎ 整数型の記号定数を使います。

- ① `CONST ic%=9` では ic% を 9 に固定します。% は整数型を示します。
- ② `FOR i%=1 TO ic%`


```
FOR j%=1 TO ic%
```

```
)
```

によって i% を 1 から 9, j% を 1 から 9 までループします。くり返しは II 編 5 章で扱います。

- ③ 五五の表にしたいというときは `CONST ic%=5` とおすことで i% を 1 から 5, j% を 1 から 5 までのループとします。このように記号定数を使うことで訂正が簡単にいくことがあります。特に ic% をプログラムの中でたくさん使っている場合それをさがすのは大変です。

【プログラム例】

```
CONST ic% = 9 ————— ic% を 9 に固定する
FOR i% = 1 TO ic% ————— i を 1 から 9 までループ
  FOR j% = 1 TO ic% ————— j を 1 から 9 までループ
    PRINT USING "## "; i% * j%; ————— i*j を表示
  NEXT j% —————
PRINT ————— 改行
NEXT i% —————
END
```

【結 果】

1	2	3	4	5	6	7	8	9
2	4	6	8	10	12	14	16	18
3	6	9	12	15	18	21	24	27
4	8	12	16	20	24	28	32	36
5	10	15	20	25	30	35	40	45
6	12	18	24	30	36	42	48	54
7	14	21	28	35	42	49	56	63
8	16	24	32	40	48	56	64	72
9	18	27	36	45	54	63	72	81

【例題 13】 文字型記号定数

m\$ を文字定数 "Microsoft" で固定し、m\$ と "Quick C", m\$ と "Quick BASIC" を表示するプログラムをつくれ。

【解 説】 ◎ 文字型の記号定数を使います。

- ① 次の形で変数 m\$ を "Microsoft" に固定します。

```
CONST m$ = "Microsoft"
```

【プログラム例】

```
CONST m$ = "Microsoft" ————— 文字定数 "Microsoft" を m$ に固定する
m1$ = m$ + " Quick C" ————— m$ と文字列の和を m1$ と m2$ に代入
m2$ = m$ + " Quick Basic" —————
PRINT m1$ ————— 結果の表示
PRINT m2$ —————
END
```

【結 果】

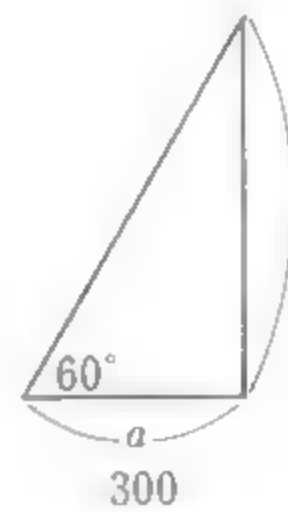
```
Microsoft Quick C
Microsoft Quick Basic
```

〔演習問題〕

- (14) x%を定数 15 として、次の 15 個のデータを読んで、その和を求めるプログラムをつくれ。

データ： 1.25, 2.56, 3.25, 4.55, 3.21
 4.55, 2.36, 3.54, 1.56, 7.51
 8.21, 9.56, 8.12, 9.21, 2.09

- (15) x!を 1.7321 として次図の l を求めるプログラムをつくれ。ただし、1.7321 は次図の a を 1 としたときの l の長さである。



- (16) a\$を"□C"として、Turbo C, Microsoft C, Quick C, Run C, Lattice C と表示するプログラムをつくれ。

〔解 答〕

(14) 【プログラム例】

```
CONST x% = 15  ----- x%を 15 定数とする
FOR i% = 1 TO x% ----- データの読み込みと和
    READ z
    zz = zz + z
NEXT i%
PRINT zz ----- 和を表示
END
DATA 1.25, 2.56, 3.25, 4.55, 3.21
DATA 4.55, 2.36, 3.54, 1.56, 7.51
DATA 8.21, 9.56, 8.12, 9.21, 2.09
```

【結 果】

71.53

(15) 【プログラム例】

```
CONST x! = 1.7321 ----- x!を定数 1.7321 とする
l = 300 * x! ----- l を求める
PRINT l ----- l の表示
END
```

【結 果】

519.63

(16) 【プログラム例】

CONST a\$ = " C"	-----	a\$を定数" C"とする
FOR i = 1 TO 5	-----	5回のくり返し
READ aa\$	-----	文字列データを読む
PRINT aa\$ + a\$	-----	読み込んだ文字列と" C"を加えて表示
NEXT i	-----	
END		
DATA Turbo, Microsoft, Quick, Run, Lattice		

【結 果】

```
Turbo C
Microsoft C
Quick C
Run C
Lattice C
```

3 章 入力・データの読み込み

INPUT, INPUT\$, INPUT#, LINEINPUT, INKEY\$,
READ, DATA, RESTORE

【例題 14】 INPUT による入力

整数を 2 つ入力し、次に実数を 2 つ、次に文字列を 2 つ入力して表示するプログラムをつくれ。

【解 説】 ◎ 数値と文字列を入力します。

- ① INPUT 変数名で変数名に値を代入します。入力するデータの型にあわせた変数名を使います。

例]

INPUT	x%	整数を 1 つ x% に入力
INPUT	x!	単精度実数を 1 つ x! に入力
INPUT	x\$	文字列を 1 つ x\$ に入力

- ② プログラムが実行されると?が表示されますのでデータを入力します。データの型が異なると指定された型で入力されます。ただし、数値型変数へ文字列の入力はできませんので”再入力して下さい”の指示がでます。

- ③ 2 つ以上のデータを入力するときは変数名をカンマで区切ります。

例]

INPUT	x%, y%	整数を 2 つ入力
INPUT	x!, y!, z!	単精度実数を 3 つ入力
INPUT	x\$, y#, z\$	文字列 1 つ、倍精度実数を 1 つ、文字列を 1 つ入力

- ④ プログラムを実行すると?が表示されますので、次のようにデータを区切って、変数の型と順番および数をそろえて入力します。

データ、データ、データ [Enter] またはデータ[]データ[]データ [Enter] ([]はスペース)

- ⑤ 型が異なったり、数が違っていたりすると再入力を促します。

【プログラム例】

```
INPUT x%, y% —— 整数 2 つ入力
INPUT x, y —— 実数 2 つ入力
INPUT x$, y$ —— 文字 2 つ入力
PRINT x%, y% —— 整数 2 つ表示
PRINT x, y —— 実数 2 つ表示
PRINT x$, y$ —— 文字 2 つ表示
END
```

【結 果】

? 25,34	———— 25, 34 
? 1.26,35.287	———— 1.26, 35.287 
? ABC,ZYX	ABC, ZYX  の例
25	34
1.26	35.287
ABC	ZYX

【例題 15】 プロンプト文付きの入力

”スウチ(1 カラ 100) ノ ニュウリョク”と表示した後に数値を入力して、表示するプログラムをつくれ。

【解 説】 ◎ プロンプト文付きで入力

- ① INPUT 変数名での入力は?が表示されるだけです。次のようにかくと” ”の中の文章を表示しますので入力する内容がわかりやすくなります。

INPUT ”文字列”;変数名

- ② PRINT ”文字列”;

INPUT 変数名

でも同じ効果がでます。

【プログラム例】

```
INPUT "1 カラ 100 ノ ニュウリョク"; x% —— 整数の入力
PRINT x% —— 表示
END
```

【結 果】

1 カラ 100 ノ ニュウリョク? 55	55 を入力の例
55	

【例題 16】 LINE INPUT による入力

文章を入力し、そのまま表示するプログラムをつくれ。

【解 説】 ◎ 文章の入力をします。

- ① LINE INPUT x\$はキーからおされた文字・文章をそのまま x\$に受けつけます。 ”や , もそのまま受けつけます。 INPUT x\$では , や ”は受けつけません。

【プログラム例】

```
LINE INPUT x$ —— 入力
PRINT x$ —— 表示
END
```

【結 果】

"Quick Basic,Quick C,Quick Fortran" 	———— 入力例
"Quick Basic,Quick C,Quick Fortran" ————	表示例 ”や , が受けつけられている

【例題 17】 INPUT\$, INPUT¥による入力

キーから 8 文字と 6 文字の 2 文字列を入力して表示するプログラムをつくれ。

【解 説】 ◎ INPUT\$と INPUT¥による指定文字数の入力をします。

① 次の形で指定した文字数の文字列を受けつけ、変数に代入します。漢字は 2 で 1 文字です。

`x$=INPUT$(8)`

入力して 8 バイト、半角文字は 8 文字
x\$に代入 漢字の場合は 4 文字

`y$=INPUT¥(6)`

6 文字、半角でも漢字でも 6 文字

② エコーバックはありません。おされたキーが指定した文字数になると変数に代入されます。␣ はしません。

【プログラム例】

```
x$ = INPUT$(8) —— 8 文字の入力
y$ = INPUT¥(6) —— 6 文字の入力
PRINT x$, y$
END
```

【結 果】

abcdefgh ijklmn abcdefghijklmn を入力 前の 8 文字が x\$, 後の 6 文字が y\$ に代入
あいうえ おかきくけこ あいうえおかきくけこを入力 前の 4 文字が x\$, 後の 6 文字が y\$ に代入

【例題 18】 キーから 1 文字入力

キーから文字がおされて入力されるのを待ち、入力されたらその文字・記号を表示するプログラムをつくれ。ただし、“e”がおされたら終了するものとする。

【解 説】 ◎ キー入力のくり返しをします。

① 次の形で 1 文字を入力し、表示します。“e”が入力されると終了します。DO 文についてはⅡ編 5 章で扱います。

DO

`x$=INKEY$` キーより 1 文字入力

`PRINT x$` 表示

`LOOP WHILE x$ <> "e"` “e”が入力されると `x$ <> "e"` が偽となり DO LOOP を終了する

【プログラム例】


```
DO
x$ = INKEY$ —— キー入力
PRINT x$; —— 表示
LOOP WHILE x$ <> "e" —— “e”のとき終了
END
```

【結 果】


54jkihkguycfe

5, 4, ..., e を入力の場合

【例題 19】 特殊キーの判定

キーをおし、 キーのときプログラムを終了し、その他のキーのときはその文字、記号を表示するプログラムをつくれ。

【解 説】 ① おされたキーを判別します。


- ① キーがおされると `x$=INKEY$` で `x$` にキーのアスキーコードが入ります。したがって、`x$` のアスキーコードをみると何がおされたかがわかります。
- ②  キーのアスキーコードは `&HOD` です。
- ③ 1F 文、GOTO 文については II 編 6 章で扱います。

【プログラム例】

```

10 : x$ = INKEY$
IF x$ = "" THEN GOTO 10
PRINT x$;
IF ASC(x$) <> &HD THEN GOTO 10
END

```

キー入力
x\$ の表示
 のとき終了

【結 果】

Acdeg153vgt fttrh gf

A c d ... f t ...  g f  とキーをおした例

【例題 20】 READ によるデータの読み込み

次のデータを配列 `r%(15)` に読み込んで和を求め表示するプログラムをつくれ。

データ: 8, 7, 2, 9, 6, 5, 4, 8, 2, 0, 5, 2, 7, 6, 7

【解 説】 ① READ によってデータを読み込みます。

- ① 次の形でデータを配列 `r%(0)~r%(14)` に読み込みます。配列については II 編 8 章、くり返しについては II 編 5 章で扱います。

```
FOR i=0 TO 14
```

```
  READ r%(i)
```

```
NEXT i
```

⋮

`r%(0)r%(1)r%(2).....r%(14)`

Data 8, 7, 2, 9, 6, 5, 4, 8

8

7

2

7

Data 2, 0, 5, 2, 7, 6, 7

- ② データ文 Data... はプログラムのどこにかいてもかまいません。前から順に変数にわりあてられます。
- ③ データ文はいくつかに分割してかくことができます。

- ④ データ数に変数名より少ないとエラーとなりますが、多いときは余った分は無視されます。

【プログラム例】

```

DIM r%(15)          配列の宣言
FOR i = 0 TO 14      データの読み込み
    READ r%(i)
NEXT i
FOR i = 0 TO 14      和を求める
    t% = t% + r%(i)
NEXT i
PRINT t%
END
DATA 8, 7, 2, 9, 6, 5, 4, 8  結果の表示
DATA 2, 0, 5, 2, 7, 6, 7

```

【結 果】

78

【例題 21】 READ によるデータの読み込み

下のデータを配列 m\$(10)に読み込み"H"で始まるものを表示するプログラムをつくれ。

データ;Tokyo, Hakodate, Morioka, Hirosaki, Hagi

Kochi, Uwajima, Kanazawa, Narita, Kisarazu

【解 説】 ◎ 文字列データを読み込みます。

- ① 変数名に文字型変数を使う他は【例題 20】とまったく同じ方法で読み込みます。
- ② $ASC(m$(i))=72$ すなわち"H"で始まる文字列を表示します。if 文についてはⅡ編 6 章で学びます。

【プログラム例】

```

DIM m$(10)          配列の宣言
FOR i = 1 TO 10      データの読み込み
    READ m$(i)
NEXT i
FOR i = 1 TO 10      "H"で始まる文字列の表示
    IF ASC(m$(i)) = 72 THEN PRINT m$(i)
NEXT i
END
DATA Tokyo, Hakodate, Morioka, Hirosaki, Hagi
DATA Kochi, Uwajima, Kanazawa, Narita, Kisarazu

```

【結 果】

Hakodate
Hirosaki
Hagi

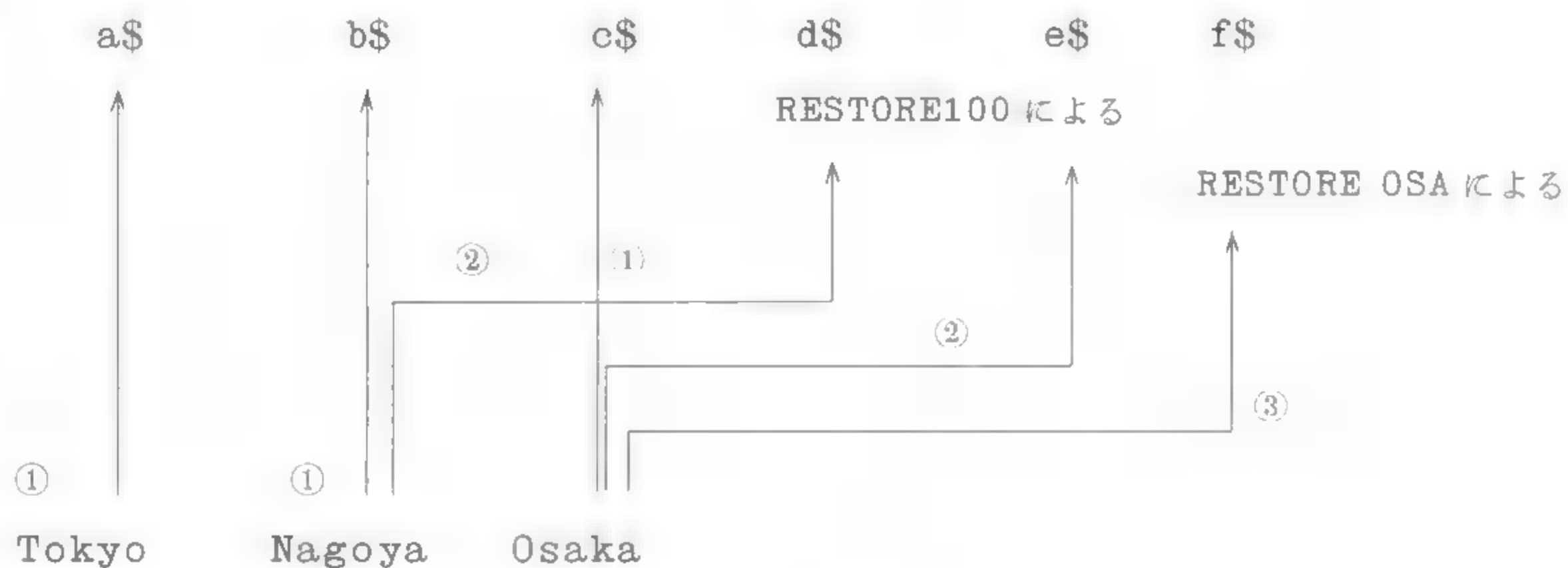
【例題 22】 データの再利用

下のデータを使って、a\$に Tokyo, b\$に Nagoya, c\$に Osaka, d\$に Nagoya, e\$に Osaka, f\$に Osaka と読み込むプログラムをつくれ。

データ;Tokyo, Nagoya, Osaka

【解 説】 ㊦ データを何回か読み込みます。

- ① READ 文では前から順にデータを読んで変数にわりあてますが、RESTORE を使って、順序をかえることができます。
- ② RESTORE 行番号または RESTORE ラベルによって、その行番号またはラベルへもどって、そこから順にデータを読み込みます。



①～③は 1 番目の READ 文～3 番目の READ 文を示す。

【プログラム例】

```

READ a$, b$, c$ ----- a$～c$にデータを読む
RESTORE 100 ----- データポインタを 100 へもどす
READ d$, e$ ----- d$, e$にデータを読む
RESTORE osa ----- データポインタを osa へもどす
READ f$ ----- f$にデータを読む
PRINT a$; " "; b$; " "; c$; " "; d$; " "; e$; " "; f$ ----- 表示
END
DATA Tokyo
100 DATA Nagoya
osa: DATA Osaka

```

【結 果】

Tokyo Nagoya Osaka Nagoya Osaka Osaka

【例題 23】 データ文に終わりのマーク

次のデータを読んで配列 x%(10)に格納し、和と平均を求めるプログラムをつくれ。ただし、データ文の最後に 0 をおき、データの終了を示すものとし、この 0 はデータには入れないものとする。

データ;7, 6, 3, 5, 9, 0

【解 説】 ◎ データ文の終わりに 0 をつけてデータの終了を示します。

- ① データ数がわかっているときは FOR～NEXT 文のくり返しなどを使って読むことができますが、データ数が不明のときはそれができません。データがないとき READ するとエラーとなります。そこで、データ文の最後に 0 などをつけて、データの終わりを示すと処理がしやすくなります。
- ② 次はその 1 例です。この形では $x(i)$ に読み込んだデータが 0 のとき読み込みを終了します。この時のデータ数は $i-1$ です。

```
FOR    i=1  TO  100
      READ x(i)
      IF x(i)=0 THEN EXIT FOR
NEXT i
```

- ③ 次の形でもデータ文の最後の 0 まで読み込みます。

```
DO While x
  c=c+1
  READ x
  x(c)=x
WEND
```

- ④ データの終わりを示すマークは 0 でなくても、-99999 などデータとして起こりえないものを使います。

【プログラム例】

FOR i = 1 TO 10	10回のくり返し
READ y%	データを読み込む
IF y% = 0 THEN EXIT FOR	データが 0 のときループの外へ分岐
x%(i) = y%	配列へデータを代入
t = t + y%	和を求める
NEXT i	
h = t / (i - 1)	平均を求める
PRINT t, h	和と平均の表示
END	
DATA 7, 6, 3, 5, 9, 0	

【結 果】

30 6

〔演習問題〕

- (17) 整数を 1 つ、文字列を 2 つ入力して順に表示するプログラムをつくれ。
- (18) プロンプト文で "ナマエ オ ニュウリョクシテクダサイ" と表示した後、na\$ に名前を入力して表示するプログラムをつくれ。
- (19) キーをおして、**[E]** のとき "end" と表示するプログラムをつくれ。
- (20) キーをおして、**[ESC]** キーのとき escape, **[TAB]** キーのとき tab と表示するプログラム

をつくれ。ただし、他のキーのときは何も表示しないで終了するものとする。

- (21) キーから 10 文字を入力して表示するプログラムをつくれ。
- (22) 次の文章をキー入力してから表示するプログラムをつくれ。ただし、”も入力するものとする。
 ”You visited Boston last year, didn't you?”
- (23) 下のデータを読んで配列 $x\%(3,7)$ に代入し、次に横の項の和を求めて、全体を表示するプログラムをつくれ。

3	2	8	6	3	4	2
8	6	7	5	2	8	7
9	4	1	7	2	6	5

- (24) 下のデータを使って、a1\$に Washington, a2\$に Chicago, a3\$に Salt Lake City, a4\$に Los Angeles, a5\$に Chicago, a6\$に Salt Lake City, a7\$に Washington, a8\$に Chicago, a9\$に Salt Lake City を代入して、順に表示するプログラムをつくれ。

データ; Washington, Chicago, Salt Lake City, Los Angeles

- (25) 次のデータを配列 $x\%(15)$ に読み、和を求めるプログラムをつくれ。ただし、0 のときデータは終了するものとする。

データ; 9, 7, 8, 5, 2, 7, 6, 8, 1, 3, 2, 4, 7, 4, 7, 0

- (26) キーがおされたら、プログラムを実行してから何秒経過したかを示すプログラムをつくれ。

〔 解 答 〕

(17) 【プログラム例】

```
INPUT x% _____ 整数の入力
INPUT x1$, x2$ _____ 文字列を 2 つ入力
PRINT x%, x1$, x2$ _____ 表示
END
```

【結 果】

```
? 555                                555
? Quick C, Quick Basic               Quick C, Quick BASIC の例
555                                Quick C
```

(18) 【プログラム例】

```
INPUT "ナメ オ ニュウリョクシテクダサイ"; na$ _____ 入力
PRINT na$ _____ 表示
END
```

【結 果】

```
ナメ オ ニュウリョクシテクダサイ? 大 山 太 郎      大山太郎 の例
大 山 太 郎
```

(19) 【プログラム例】

```

10 x$ = INKEY$ ————— 1 文字の入力
   IF x$ = "" THEN GOTO 10 —————
   IF ASC(x$) = &H45 THEN PRINT "end" ————— "E" のとき end と表示
END

```

【結 果】

```

end          "E" をおしたとき

```

(20) 【プログラム例】

```

10 x$ = INKEY$ ————— キーがおされたら x$ に代入される,
   IF x$ = "" THEN GOTO 10 ————— おされないときはくり返す
   IF ASC(x$) = &H1B THEN PRINT "escape" ————— [ESC] キーのときの表示
   IF ASC(x$) = &H9 THEN PRINT "tab" ————— [TAB] キーのときの表示
END

```

【結 果】

```

tab          [TAB] キーをおした例
escape       [ESC] キーを入力した例

```

(21) 【プログラム例】

```

x$ = INPUT$(10) ————— 10 文字の入力
PRINT x$ ————— 表示
END

```

【結 果】

```

Quick C Qu          Q.....u の例

```

(22) 【プログラム例】

```

LINE INPUT x$ ————— 文章の入力
PRINT x$ ————— 表示
END

```

【結 果】

```

"You visited Boston last year,didn't you ?" ————— キー入力
"You visited Boston last year,didn't you ?" ————— 表示

```

(23) 【プログラム例】

```

DIM x%(3, 8) ————— データを 2 次元配列へ読み込み横の和を求める
FOR i = 0 TO 2
  FOR j = 0 TO 6
    READ x%(i, j)
    x%(i, 7) = x%(i, 7) + x%(i, j) ————— 横の和を求める
  NEXT j, i

```

```

FOR i = 0 TO 2 ————— 2次元の表を表示
  FOR j = 0 TO 7
    PRINT x%(i, j);
  NEXT j
PRINT
NEXT i
END
DATA 3, 2, 8, 6, 3, 4, 2
DATA 8, 6, 7, 5, 2, 8, 7
DATA 9, 4, 1, 7, 2, 6, 5

```

【結 果】

```

3  2  8  6  3  4  2  28
8  6  7  5  2  8  7  43
9  4  1  7  2  6  5  34

```

(24) 【プログラム例】

```

READ a1$, a2$, a3$, a4$ ————— a 1$~a 4$にデータを読み込み
RESTORE toshi ————— toshi ヘデータポインタをもどす
READ a5$, a6$ ————— a 5$, a 6$の読み込み
RESTORE ————— 先頭ヘデータポインタをもどす
READ a7$, a8$, a9$ ————— a 7$~a 9$の読み込み
PRINT a1$; " "; a2$; " "; a3$ ————— 結果の表示
PRINT a4$; " "; a5$; " "; a6$
PRINT a7$; " "; a8$; " "; a9$
END
DATA Washington
toshi: DATA Chicago, Salt Lake City, Los Angeles

```

【結 果】

```

Washington Chicago Salt Lake City
Los Angeles Chicago Salt Lake City
Washington Chicago Salt Lake City

```

(25) 【プログラム例】

```

DIM x%(15)
FOR i = 0 TO 14 ————— 15回のくり返し
  READ y% ————— データの読み込み
  IF y% = 0 THEN EXIT FOR ————— データが0のときループの外へ分岐
  x%(i) = y% ————— 配列ヘデータを代入
  t = t + y% ————— 和を求める
NEXT i
PRINT t ————— 和の表示
END
DATA 9, 7, 8, 5, 2, 7, 6, 8
DATA 1, 3, 2, 4, 7, 4, 7, 0

```

【結 果】

80

(26) 【プログラム例】

```
x1 = TIMER ----- 現在の秒を代入
DO ----- キーがおされるのを待つ
LOOP WHILE INKEY$ = "" -----
x2 = TIMER ----- 現在の秒を代入
PRINT x2 - x1 ----- 経過時間を表示
END
```

【結 果】

15

4 章 書式指定の表示と印刷

PRINT USING, LPRINT USING

【例題 24】 文字の書式指定表示

a\$を"Quick Basic"として a\$を①12 桁で、②"ゲンゴ□ハ□"の後に 15 桁で、③先頭 1 文字を、④5 桁を表示し、次に 3 スペースの後に a\$全体を表示するプログラムをつくれ。

【解 説】 ◎ 文字列を書式指定で表示します。

① PRINT USING の後に書式を指定します。

```
PRINT USING "&[ ] [ ] [ ] [ ]&";a$
```

6 桁

& & でかこむと && を含む桁数で a\$を表示。
この場合 6 桁です。文字は左づめです。

```
PRINT USING "ゲ`ンゴ`ハ`&`&";a$
```

文字はそのまま表示

桁数が足りないときは && で指示された桁のみ表示

```
PRINT USING "!";a$
```

先頭 1 文字を表示

```
PRINT USING "@";a$
```

桁数に関係なく文字列を表示

② PRINT を LPRINT とすると印刷されます。

【プログラム例】

```
a$ = "Quick Basic"
PRINT USING "&          &"; a$ —— 12 桁で表示
PRINT USING "ゲ`ンゴ`ハ`&`&"; a$ —— 文字の後に表示
PRINT USING "!"; a$ —— 先頭 1 文字の表示
PRINT USING "&  &  @"; a$; a$ —— 5 文字と全体を表示
END
```

【結 果】

```
Quick Basic
ゲ`ンゴ`ハ Quick Basic
Q
Quick    Quick Basic
```


【例題 25】 数値の書式指定

$x\%$ を12345、 $y\%$ を1234.56、 $z\%$ を-9876として① $x\%$ と $z\%$ を6桁で、② $y\%$ を全体を7桁小数点以下2桁と全体を6桁小数点以下1桁で、③ $x\%$ を全体6桁で先頭と最後に+をつけて、④ $z\%$ を全体を7桁で先頭と最後に-をつけて表示するプログラムをつくれ。

【解 説】 ◎ 数値の書式指定をします。

- ① 数値の書式指定も PRINT USING の後に書式をかきます。
 ② 次のような指定となります。

PRINT USING "##### [] [] #####"; $x\%$; $z\%$

全体を6桁先頭がプラスのとき省略右詰め

全体を6桁 マイナスのとき

-がつく. 符号も含めて6桁

PRINT USING "#####.## [] #####.#" ; $y\%$; $y\%$

全体7桁小数点以下2桁

全体6桁で小数点以下1桁四捨五入される

PRINT USING "+##### #####+" ; $x\%$; $x\%$

プラスのとき先頭に+がつく

プラスのとき最後に+がつく

PRINT USING "##-"; $z\%$

マイナスのとき最後に-がつく

【プログラム例】

```

 $x\%$  = 12345
 $y\%$  = 1234.56
 $z\%$  = -9876
PRINT USING "##### #####";  $x\%$ ;  $z\%$ 
PRINT USING "#####.## #####. #";  $y\%$ ;  $y\%$ 
PRINT USING "+##### #####+" ;  $x\%$ ;  $x\%$ 
PRINT USING "#####-";  $z\%$ ;  $z\%$ 
END

```

値の代入

6桁で表示

小数点以下2桁と1桁で表示

+を先頭と最後につけて表示

-を先頭と最後につけて表示

【結 果】

12345	-9876
1234.56	1234.6
+12345	12345+
-9876	9876-

【例題 26】 *と¥, カンマ付き表示

x%を12345として①全体を8桁で、8桁未満のときは先頭を*をうめて、②全体を9桁で、先頭に¥をつけ、小数点以下2桁で、③全体を9桁で先頭に¥をつけ、9桁未満のときは*でうめて、④全体を8桁で3桁のカンマ付きで表示するプログラムをつくれ。ただし、桁数には*, ¥, ., ,も含めるものとする。

【解 説】 ◎ *, ¥, カンマをつけます。

- ① 次の書式で*をつけます。

```
PRINT USING "*****";x%
```

全体8桁、8桁未満のときは先頭を*でうめる。

- ② 次の書式で¥をつけます。

```
PRINT USING "¥¥####.##";x%
```

全体9桁、先頭に¥がつく

- ③ 次の書式で¥をつけます。桁数が少ないときは先頭を*でうめます。

```
PRINT USING "**¥#####";x%
```

全体9桁、先頭に¥がつく
9桁未満のときは*でうめる。

- ④ 次の書式で3桁ごとの,をつけます。

```
PRINT USING "####,###";x%
```

全体8桁
3桁ごとのカンマ付き

【プログラム例】

```
x% = 12345
PRINT USING "*****"; x% —— *でうめて表示
PRINT USING "¥¥####.##"; x% —— ¥つきで表示
PRINT USING "**¥#####"; x% —— ¥つき*でうめて表示
PRINT USING "####,###"; x% —— 3桁ごとのカンマ付き表示
END
```

【結 果】

```
***12345
¥12345.00
***¥12345
12,345
```

【例題 27】 指数書式指定表示

$x!$ を 12.345 として①0.---E--の形の指数で、②1.---E--の形の指数で、③値の前に+符号をつけた指数とEの後を3桁とした指数で、④先頭に#,最後に_ (アンダスコア) をつけて、⑤全体を####の指定で、ただし、桁不足のときは先頭に%をつけて、各々表示するプログラムをつくれ。

【解 説】 ◎ 指数書式の指定をします。

① 次の書式で指数の書式を指定します。

```
PRINT USING "#.###^"; x!
```

指数、小数点以上なし
小数点以下3桁、指数部は
E+02のように4桁

```
PRINT USING "#.###^"; x!
```

指数、指数部は
E+001のように5桁

② 次の書式は特殊なものです。

```
PRINT USING "_###.###_"; x!
```

→ _が1つ表示される
→ _の後の1文字はそのまま表示、#が表示される。
この#は桁を示す#ではない

```
PRINT USING "%####"; x!
```

→ 桁数が足りないとき%を表示して示す

【プログラム例】

```
x! = 12.345
PRINT USING "#.###^"; x! ----- 指数表示、指数部4桁
PRINT USING "##.##^"; x! ----- 指数表示、指数部4桁
PRINT USING "+#.###^"; x! ----- 指数表示、指数部4桁と5桁
PRINT USING "_###.###_"; x! ----- #と_を表示
PRINT USING "%####"; x! ----- %で桁不足を表示
END
```

【結 果】

```
0.123E+02
1.23E+01
+1.235E+01 +1.235E+001
#12.345_
% 12
```

【例題 28】 書式指定で表を表示

結果のように次のデータを名前を 11 桁 2 桁あけて獲得賞金額を 11 桁で 3 桁ごとのカンマ付き、1 桁あけて年齢を 3 桁で表示するプログラムをつくれ。

データ：名前・獲得賞金・年齢の項とする

```

エタ` キミコ,68123415,36,タナカ ミチヨ,456365,29
イシ`マ カス`コ,712568,33,ウエタ` ノリコ,5328756,31
テシ`マ キヨミ,9563255,34,ナカムラ サチコ,658599,28

```

【解 説】 ◎ 書式指定で表を表示

- ① 文字列、数値などを表示するとき書式指定をしないと、表示開始桁位置や最後の桁がばらばらで表になりません。
- ② 今回は文字を 11 桁、2 スペースあけて数値を 11 桁で 3 桁ごとのカンマ付き、1 スペースあけて 3 桁で表示します。

【プログラム例】

```

FOR i = 1 TO 6
    READ a$, x%, y%
    PRINT USING "&          & #####,### ###"; a$; x%; y%
NEXT i
END
DATA エタ` キミコ,68123415,36,タナカ ミチヨ,456365,29
DATA イシ`マ カス`コ,712568,33,ウエタ` ノリコ,5328756,31
DATA テシ`マ キヨミ,9563255,34,ナカムラ サチコ,658599,28

```

データを読んで表示

【結 果】

エタ` キミコ	68,123,415	36
タナカ ミチヨ	456,365	29
イシ`マ カス`コ	712,568	33
ウエタ` ノリコ	5,328,756	31
テシ`マ キヨミ	9,563,255	34
ナカムラ サチコ	658,599	28

〔演習問題〕

- (27) a\$を"Quick", b\$を"Basic"としてa\$を8桁、b\$を8桁で続けて表示し、次にa\$とb\$の先頭1文字を続けて表示するプログラムをつくれ。
- (28) x%を3568, y!を152.124として、①x%を5桁、y!を7桁、小数点以下2桁で、②x%を6桁小数点以下1桁、y!を6桁小数点以下1桁で、③x%を5桁で+を先頭につけて、y!を9桁で小数点以下3桁で+を最後につけて表示するプログラムをつくれ。
- (29) x!を13.259, y!を22.458としてz!をその積とし、式と結果を表示するプログラムをつくれ。ただし、①x!とy!は全体6桁、小数点以下3桁、結果は全体7桁、小数点以下3桁、②全体を指数で表示するものとする。

- (30) 次のデータを a%, b%, c% によんで書式指定を使って, 1989 ネン 5 ガツ 15 ニチ と表示するプログラムをつくれ.

データ: 1989, 5, 15

- (31) 次のデータを表にするプログラムをつくれ. 項目名として"ショウヒン", "タンカ", "コスウ", "キングク"をつけ, 金額には先頭に"¥", 個数には後に"コ"をつけるものとする.

データ: ハソコン, 125000, 25, ワープロ, 73500, 110
 フリタ, 68700, 25, ディスプレイ, 98500, 35
 フロッタ, 146000, 58, フロッピー, 145, 3560

〔解 答〕

- (27) 【プログラム例】

```
a$ = "Quick"
b$ = "Basic"
PRINT USING "&    &&    &"; a$; b$ —— 表示
PRINT USING "!"; a$; b$ —— 先頭文字の表示
END
```

【結 果】

```
Quick    Basic
QB
```

- (28) 【プログラム例】

```
x% = 3568
y! = 152.124
PRINT USING "#####.##"; x%; y! —— 5桁と小数点以下2桁の表示
PRINT USING "#####.#"; x%; y! —— 小数点以下1桁の表示
PRINT USING "+#####.###+"; x%; y! —— +付き表示
END
```

【結 果】

```
3568    152.12
3568.0   152.1
+3568    152.124+
```

- (29) 【プログラム例】

```
x! = 13.259 —— 代入
y! = 22.458 ——
z! = x! * y! —— 積
PRINT USING "##.###*##.###=##.###"; x!; y!; z! —— 表示
PRINT USING "##.###^*##.###^=##.###^"; x!; y!; z! —— 指数表示
END
```

【結 果】

```
13.259*22.458=297.771
1.326E+01* 2.246E+01= 2.978E+02
```


(30) 【プログラム例】

```

READ a%, b%, c%
PRINT USING "#### ネン # カツ ## ニチ"; a%; b%; c% —— 表示
END
DATA 1989, 5, 15

```

【結 果】

1989 ネン 5 カツ 15 ニチ

(31) 【プログラム例】

```

PRINT USING "&      &      ■      &      &      &      &"; "ショウヒン"; "タンカ"; "コスウ"; "キンカク"
FOR i = 1 TO 6
    READ a$, x!, y%
    z& = x! * y%
    PRINT USING "&      ■      YY###,###      ####      YY##### ,###"; a$; x!; y%; z&
NEXT i
END
DATA ハ`ソコン, 125000, 25, ワ-プ`ロ, 73500, 110
DATA プ`リンタ, 68700, 25, デ`イスプ`レイ, 98500, 35
DATA プ`ロッタ, 146000, 58, フ`ロッピ`イ, 145, 3560

```

表題
データを読んで
表示

【結 果】

ショウヒン	タンカ	コスウ	キンカク
ハ`ソコン	¥125,000	25 コ	¥3,125,000
ワ-プ`ロ	¥73,500	110 コ	¥8,085,000
プ`リンタ	¥68,700	25 コ	¥1,717,500
デ`イスプ`レイ	¥98,500	35 コ	¥3,447,500
プ`ロッタ	¥146,000	58 コ	¥8,468,000
フ`ロッピ`イ	¥145	3560 コ	¥516,200

5章 くり返し

FOR～NEXT, WHILE～WEND, DO～LOOP
WHILE, DO～LOOP UNTIL, DO WHILE～
LOOP, DO UNTIL～LOOP, EXIT FOR, EXIT DO

【例題 29】 FOR～NEXT による回数指定のくり返し

くり返しを使って ABC…Z を表示するプログラムをつくれ。

【解 説】 ◎ FOR～NEXT による回数指定のくり返しをします。

- ① 次の書式で指定回数のくり返しを行います。

```
FOR 変数=初期値 TO 終値 STEP 増分  
  処理  
NEXT 変数
```

- ② 例として FOR i=5 TO 82 STEP 7 ~ NEXT i とするとまず i の値を 5 とし、処理を行います。次に i に 7 を加え i の値は 12 となります。終値に達していなければ再び処理を行います。同様に i の値を 5, 12, 19, 26, 33, 40, 47, 54, 61, 68, 75, 82, 89 となります。i の値が 89 と終値をこえるとくり返しを終了します。
- ③ 初期値>終値のときは増分をマイナスとすれば変数の値を毎回減少させてくり返しを行います。たとえば、FOR i=30 TO 10 STEP -5 処理 NEXT i では i の値を 30, 25, 20, 15, 10 としてくり返しをします。この場合は i の値が 5 となり、終値より小さくなるとくり返しを終了します。
- ④ NEXT i の部分は FOR の中で使っている変数名と同じものまたは省略します。したがって、例では NEXT i または NEXT です。
- ⑤ FOR i=500 TO 1000 STEP -50 のように論理が成立しないときは、1 回もくり返しは行われません。
- ⑥ 本題では ABC…Z の表示ですから i の値に 65, 66, 67…90 をつくって、そのアスキーコードを表示すればよいことになります。この例のように変数の値を処理の中で使うこともできますし、使わなくてもかまいません。

【プログラム例】

```
FOR i = 65 TO 90 — i が 65 から 90 までくり返し  
  PRINT CHR$(i); — i をアスキーコードとする文字を表示  
NEXT i  
END
```

【結 果】

ABCDEFGHIJKLMNOPQRSTUVWXYZ

【例題 30】 WHILE～WEND によるくり返し

整数を入力し、順に表示するプログラムをつくれ。ただし、0 が入力されたら 0 を表示して終わるものとする。

【解 説】 ◎ WHILE～WEND によるくり返しをします。

- ① WHILE 条件～WEND は条件が成立している間～の処理を行います。
- ② したがって、例解では WHILE x すなわち x が真の間くり返します。くり返しの内容は x の入力とその表示です。0 以外の入力ときはそれをくり返し、0 が入力されると 0 を表示します。次に条件が偽となりますから終了です。
- ③ 最初に x=1 としておかないと WHILE x が偽となって処理を 1 回もせずに終わります。
- ④ WHILE 1 ～ WEND では条件が永久に真ですから、無限のくり返しとなります。後でみるように、途中から if 文などを使ってぬけでることができます。

【プログラム例】

```

x = 1 ————— x を 1 とする。次の WHILE 条件を真とする目的
WHILE x ————— x が真の間くり返し
    INPUT x
    PRINT x
WEND
END

```

【結 果】

```

? 5          5, 2, 45, -87, 26, 0 を入力の例
5
? 2
2
? 45
45
? -87
-87
? 26
26
? 0
0

```

【例題 31】 DO～LOOP WHILE によるくり返し

次のデータを読んで 0 まで表示するプログラムをつくれ。

データ; 5, 2, 4, 8, 2, 9, 0, 1, 4

【解 説】 ◎ DO～LOOP WHILE によるくり返しをします。

- ① Quick BASIC は DO～LOOP という汎用くり返しを準備しています。これは 4 つの変形を持っています。

	くり返しの条件	条件判断のタイミング
DO~LOOP WHILE 条件	……条件が成立している間	処理の後
DO~LOOP UNTIL 条件	……条件が成立するまで	処理の後
DO WHILE 条件~LOOP	……条件が成立している間	処理の前
DO UNTIL 条件~LOOP	……条件が成立するまで	処理の前

- ② 本例は DO 処理 WHILE $x \neq 0$ です。処理を行ってから、 x が 0 でなければくり返します。データが 0 の場合は x に 0 を読み、 x を表示した後、 x が 0 となり条件が偽となって終了します。

【プログラム例】

```

DO
  READ x
  PRINT x;
LOOP WHILE x <> 0
END
DATA 5, 2, 4, 8, 2, 9, 0, 1, 4

```

くり返し
データを読んで表示
 x が 0 でないときくり返し

【結果】

5 2 4 8 2 9 0

【例題 32】 DO~LOOP UNTIL

数 x を入力して、 x の階乗を求めるプログラムをつくれ。

【解説】 ◎ DO~LOOP UNTIL のくり返しをします。

- ① DO~LOOP UNTIL 条件では条件が成立するまでくり返しをします。条件の判断は処理をすませた後です。
- ② 本例での t 、 y の動きは次のようになります。

	t (階乗の値)	y
初期値	1	5 (入力した値)
1回目	5 (1 * 5)	4 (5 - 1)
2回目	20 (5 * 4)	3 (4 - 1)
3回目	60 (20 * 3)	2 (3 - 1)
4回目	120 (60 * 2)	1 (2 - 1)
5回目	120 (120 * 1)	0 (1 - 1)

y の値が 0 となっておりくり返しを終了

- ③ このプログラムでは x に 0 を入力すると $y = y - 1$ で y の値は -1 となり、次は -2, -3, ... となります。したがって、 y が 0 になりませんから無限にくり返しが続きます。積 $t = t * y$ が最初に $t = 0$ となるため t の値は 0 です。
- ④ また、 x に実数で小数点のつく値をいれると y の値が 0 になりませんからこれも無限にくり返されます。ただし、この場合は t の値はオーバーフローして終了します。たとえば、 x に 5.3 を入力すると t の値は $1 * 5.3 * 4.3 * 3.3 * 2.3 * 1.3 * 0.3 * -0.7 * -1.7 * -2.7 * \dots$ となります。

【プログラム例】

```

t = 1 ————— 階乗の入る変数の初期値
INPUT x ————— x の入力
y = x ————— x の値を y に代入
DO ————— くり返し
    t = t * y ————— 積 1*y*(y-1)*(y-2)…
    y = y - 1 ————— y-1, y-2, y-3, …をつくる
LOOP UNTIL y = 0 ————— y が 0 になるとくり返しを終了
PRINT x; "!="; t ————— 結果の表示
END

```

【結果】

```

? 10 ————— 10 を入力の例
10 != 3628800

```

【例題 33】 DO UNTIL~LOOP

a, b, c さんの成績は各々 60, 80, 50 点とする。次に最低点 x を入力し、 x が 40 点未満のとき、1 回に 10 点のげたをはかせる。この時 1 回につき a は 3 点, b は 2 点, c は 4 点のげたを上のをせるものとする。最低点がげたを何回かはいて 40 点以上となったときの x , a, b, c の点を表示するプログラムをつくれ。

【解説】 ㉓ DO UNTIL~LOOP によるくり返しをします。

- ① DO UNTIL 条件~LOOP では条件が成立するまで処理をくり返します。
- ② 処理の結果で条件を判断する場合は DO UNTIL~LOOP も DO~LOOP UNTIL も同じです。ただし、DO~LOOP へ入る前の値で条件が決まる（本例は入力された x の値で条件が決まる）ときはこの 2 つは意味が違います。

$x=10$	$x=10$
DO UNTIL $x \geq 5$	DO
処理	処理
LOOP	LOOP UNTIL $x \geq 5$
x が 10 なので $x \geq 5$ の条件が成立するのでループ内の処理は 1 回もしないで LOOP の次へ進む	x が 10 でも処理を 1 回行って $x \geq 5$ が成立するので LOOP の次へ進む

- ③ x の点を 25 とすると次のような処理となります。

	初期値	1 回目の処理	2 回目の処理
a	60	63	66
b	80	82	84
c	50	54	58
x	25	35	45 ($x \geq 40$ のため終了)

【プログラム例】

```

a = 60
b = 80
c = 50
INPUT x
DO UNTIL x >= 40
    x = x + 10
    a = a + 3
    b = b + 2
    c = c + 4
LOOP
PRINT "x="; x, "a="; a
PRINT "b="; b, "c="; c
END

```

a, b, c の初期値
 x の入力
 x >= 40 になるまでくり返し
 x に 10 を加える
 a に 3 を加える
 b に 2 を加える
 c に 4 を加える
 x, a, b, c の表示

【結 果】

? 25		x=25 を入力の例
x= 45	a= 66	
b= 84	c= 58	
? 50		x=50 を入力の例
x= 50	a= 60	
b= 80	c= 50	
? 35		x=35 を入力の例
x= 45	a= 63	
b= 82	c= 54	

【例題 34】 DO WHILE~LOOP

数 x を入力して 2^x を求めるプログラムをつくれ。

【解 説】 ◎ DO WHILE~LOOP によるくり返しをします。

- ① DO WHILE 条件~LOOP は処理の前に条件を判断し、条件が成立しているとき処理を行い、くり返しをします。条件が成立しないときは LOOP の外へ出ます。したがって、最初に条件が成立しないと、処理は一度も実行されません。
- ② 本例では x に整数を入力し、その回数だけ 2 を掛けます。整数から順に 1 をひいていって、値が 0 になるとくり返しの終了です。
- ③ x に 0 が入力されると条件が不成立のため、処理はされずもとの $z\%=1$ が生きていて、 2^0 が 1 となります。

【プログラム例】

```

z% = 1
INPUT x%
y% = x%
DO WHILE y% <> 0
    z% = z% * 2
    y% = y% - 1
LOOP
PRINT "2^"; x%; "="; z%
END

```

積の入る変数の初期値
 x の入力
 y に x を代入
 くり返し、y が 0 でないときくり返す
 2^x を求める
 結果の表示

【結 果】

? 6
2^ 6 = 64

6 を入力の場合

? 0
2^ 0 = 1

0 を入力の場合

【例題 35】 FOR～NEXT のネスティング

上段に横に 123…10 とかき、縦に 123…10 とかき、その交点に横の値と縦の値の積をかく
10*10 の表をつくるプログラムをつくれ.

1	2	3	4	10
2	4	6	8		20
3	↑				12 ← 4 と 3 の積
4	2 と 2 の積				
⋮					
⋮					
10	20	30	40		100

10 と 10 の積

【解 説】 ◎ FOR～NEXT 文のネスティングをします.

① 次の形で 2 重のくり返し (ネスティング) をします.

```

FOR i=1 TO 10
  FOR j=1 TO 10
    処 理
  NEXT j
NEXT i

```

i を 1 から 10 までくり返す
 i を 1 とし j を 1, 2, 3, ...10
 i を 2 とし j を 1, 2, 3, ...10
 i を 3 とし j を 1, 2, 3, ...10
 ⋮
 i を 10 とし j を 1, 2, 3, ...10
 とする

② ネスティングは何重になってもかまいません. ただし, 中の FOR～NEXT 文は外の FOR～NEXT の中に完全に入っていなければなりません.

正しい例

```

FOR i=1 TO 5
  FOR j=5 TO 50
    FOR k=20 TO 100
      処 理
    NEXT k
  NEXT j
NEXT i

```

正しくない例

```

FOR i=1 TO 5
  FOR j=5 TO 50
    FOR k=20 TO 100
      処 理
    NEXT j
  NEXT k
NEXT i

```

- ③ FOR～NEXT 文がネスティングされ、NEXT が並ぶような場合は次のようにすることができます。

下の 3 例はいずれも正しい使い方です。

```

FOR i=1 TO 10
    FOR j=3 TO 8
        処 理
    NEXT j
NEXT i

```

```

FOR i=1 TO 10
    FOR j=3 TO 8
        処 理
    NEXT
NEXT

```

```

FOR i=1 TO 10
    FOR j=3 TO 8
        処 理
    NEXT j, i

```

【プログラム例】

```

FOR i = 1 TO 10
    FOR j = 1 TO 10
        PRINT USING "####"; i * j;
    NEXT j
    PRINT
NEXT i
END

```

i=1 から 10 までくり返し
 j=1 から 10 までくり返し
 i*j の表示
 1*1, 1*2, 1*3, ... 1*10
 2*1, 2*2, 2*3, ... 2*10
 ...
 10*1, 10*2, 10*3, ... 10*10 を表示

【結 果】

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

【例題 36】 FOR～NEXT からの脱出

1+2+3+...+i が 1000 をこえる最初の i の値とそのときの和を求めるプログラムをつくれ。

【解 説】 ◎ EXIT FOR は FOR～NEXT ループの途中で脱出します。

- ① FOR i=初期値 TO 終値～NEXT 文の中で i が終値になる前に別の条件でくり返しを終わらせたいことがあります。そのために IF 条件 THEN EXIT FOR という形が使えます。EXIT FOR は NEXT 文の次へ進みます。
- ② 本例では 1+2+... の和が t に求められます。この和 t が 1000 をこえたとき EXIT FOR でくり返しから脱出します。
- ③ i の終値は充分大きくして IF 文の条件によるループの脱出より先にくり返しが終わらないように

します。

- ④ 従来の BASIC では GOTO 文あるいは GOSUB 文でループの外へ脱出しました。

新しい形

```
FOR i=1 TO 100
  処 理
  IF 条件 THEN EXIT FOR
NEXT i
```

→
条件が成立するとここへ分岐する

従来の形

```
10 FOR i=1 TO 100
  / 処 理
80 IF 条件 THEN GOTO 100
90 NEXT i
```

100 ←
条件が成立するとここへ分岐する

【プログラム例】

```
t = 0 ————— 和の入る変数 t の初期値を 0 とする
FOR i = 1 TO 100 ————— i を 1 から 100 までくり返し
  t = t + i ————— t に 1+2+... を求める
  IF t > 1000 THEN EXIT FOR ————— t が 1000 をこえると脱出
NEXT i —————
PRINT "1+2+3+...+"; i; "="; t ————— 結果の表示
END
```

【結 果】

1+2+3+...+ 45 = 1035

【例題 37】 EXIT DO による脱出

数 x を入力して表示するプログラムをつくれ。ただし、 x がマイナスのとき、その値を表示しないで終了するものとする。

【解 説】 ◎ EXIT DO で DO～LOOP ループの途中から脱出します。

- ① DO～LOOP の中に IF 条件 THEN EXIT DO を入れると条件が成立したとき DO～LOOP から脱出できます。
- ② 本例では次のようになります。

x に 0 以上の値が入力されている間は DO UNTIL $x < 0$ の条件が成立しませんから入力と表示を

くり返します。

x にマイナス値が入力されると `IF $x < 0$ THEN EXIT DO` によって、終了します。つまり表示をしないで終了します。この `IF` 文がないと、マイナスの x の値を表示した後 `DO UNTIL $x < 0$` の条件が成立して終了します。

この例では `IF` 文は x がマイナスのとき、その値を表示するかしないかの違いです。

【プログラム例】

<code>DO UNTIL $x < 0$</code>	—	x がマイナスになるまでくり返し
<code>INPUT x</code>	—	x の入力
<code>IF $x < 0$ THEN EXIT DO</code>	—	x がマイナスのとき LOOP の次へ脱出
<code>PRINT x</code>	—	x の表示
<code>LOOP</code>	—	
<code>END</code>		

【結 果】

<code>? 110</code>	110, 25, -86 を入力の例
110	
<code>? 25</code>	
25	
<code>? -86</code>	

【例題 38】 EXIT DO による DO~LOOP UNTIL からの脱出

アルファベット大文字 A~J を順に表示するプログラムをつくれ。

【解 説】 ◎ EXIT DO によって DO~LOOP から脱出します。

- ① DO~LOOP の処理の中に `IF 条件 THEN EXIT DO` を入れると条件が成立したとき DO~LOOP の外へ脱出できます。
- ② 本例では DO~LOOP 条件の条件は A, B, C... と表示して行って Z になったときです。しかし、途中で `IF $t \geq 75$ THEN EXIT DO` とします。すなわち、 t が 75 (J のアスキーコード) のとき EXIT DO でループの外へ脱出します。すなわち、A~J を表示して終了します。

【プログラム例】

<code>t = 64</code>	—	t の初期値を 64 とする
<code>DO</code>	—	くり返し、 <code>x\$</code> が "Z" で終了
<code>t = t + 1</code>	—	t に 1 を加え 65, 66, 67... をつくる, A, B, C... のアスキーコード
<code>IF $t \geq 75$ THEN EXIT DO</code>	—	t が 75 以上になったらループの外へ脱出
<code>PRINT CHR\$(t);</code>	—	表示
<code>LOOP UNTIL $x\$ = "Z"$</code>	—	
<code>END</code>		

【結 果】

ABCDEFGHIJ

〔演習問題〕

- (32) アルファベット小文字で z から a まで 1 行に表示するプログラムをつくれ。
- (33) "アイウエオカキクケコ" を 1 桁おきに 1 行に表示するプログラムをつくれ。
- (34) WHILE~WEND を使って "ABC...Z" を表示するプログラムをつくれ。
- (35) 1 桁の整数の乱数をつくり表示するプログラムをつくれ。ただし、0 のとき表示して終了するものとする。
- (36) 下のデータを読み、1 行に表示するプログラムをつくれ。ただし、"E" のとき、それを表示して終了するものとする。
データ; T, O, K, Y, O, E, H, I, M, E
- (37) $1^2+2^2+3^2+\dots+i^2$ が 5000 をこえる最初の和を求めるプログラムをつくれ。
- (38) 下のデータを読み表示するプログラムをつくれ。ただし、0 の前までのデータとする。
データ; 5, 2, 3, 7, 8, 1, 2, 3, 9, 1, 3, 8, 1, 7, 1,
0, 5, 2, 9, 9
- (39) DO UNTIL~LOOP を使い、x にマイナスの値が入力されるまで値を入力し、表示をくり返すプログラムをつくれ。
- (40) 数 x を入力し、 x^2 を表示するプログラムをつくれ。 x^2 が 10000 以上となったとき終了するものとする。
- (41) 1000 から下のデータを順にマイナスして、0 以下となったときの値を求めるプログラムをつくれ。
データ; 120, 52, 456, 85, 92, 140, 358
- (42) $1+2+3+\dots+t$ を行い和が 10000 をこえる最小の t の値と和を求めるプログラムをつくれ。

〔解 答〕

(32) 【プログラム例】

```
FOR i = 122 TO 97 STEP -1
    PRINT CHR$(i);
NEXT i
END
```

i が 122 から 97 までくり返し
i の値をアスキーコードとする文字を表示

【結 果】

zyxwvutsrqponmlkjihgfedcba

(33) 【プログラム例】

```
FOR i = 177 TO 186
    PRINT CHR$(i); " ";
NEXT i
END
```

i が 177 から 186 までくり返し
i の値をアスキーコードとする文字を表示

【結 果】

ア イ ウ エ オ カ キ ク ケ コ

(34) 【プログラム例】

```

x = 1 ————— くり返しの条件を成立させるため x=1 とする
y = 65 ————— y の初期値 "A" のアスキーコード
WHILE x ————— x が 0 以外るときくり返し
    PRINT CHR$(y); — y をアスキーコードとする文字の表示
    y = y + 1 ————— y を 65, 66, 67, ... 91 とする
    x = 91 - y ————— x を 91-y とする. y が 90 で z を表示した後,
WEND ————— y=y+1 により y が 91 となり, x=91-y により
END ————— x が 0 となって終了する

```

【結 果】

ABCDEFGHIJKLMNOPQRSTUVWXYZ

(35) 【プログラム例】

```

x = 1 ————— くり返しの条件を成立させるため x=1 とする
WHILE x ————— x が 0 以外るときくり返し, 乱数の値が 0 となると終了
    x = INT(RND * 10) — x に 1 桁の乱数を代入
    PRINT x; ————— x の表示
WEND
END

```

【結 果】

7 5 5 2 3 7 0

(36) 【プログラム例】

```

x = 1 ————— くり返しの条件を成立させるため x=1 とする
WHILE x ————— x が 0 以外るときくり返し, x$が"E"のとき x が 0 で終了
    READ x$ ————— 1 文字ずつ読み x$に代入
    PRINT x$; ————— x$の表示
    x = ASC(x$) - 69 — 文字のアスキーコード-69 を x に代入. "E"のアスキーコ
WEND ————— ードが 69 のため"E"のとき x が 0 となる.
END
DATA T, O, K, Y, O, E, H, I, M, E

```

【結 果】

TOKYOE

(37) 【プログラム例】

```

t = 0 ————— i の初期値を 0 とする. i は和が入る
FOR i = 1 TO 100 ————— i を 1 から 100 までくり返す
    t = t + i * i ————— i2 の和を求める
    IF t > 5000 THEN EXIT FOR — 和が 5000 をこえたら終了
NEXT i
PRINT "1^2+2^2+3^2+.....+"; i; "^2="; t ————— 結果の表示
END

```

【結 果】

$1^2 + 2^2 + 3^2 + \dots + 25^2 = 5525$

(38) 【プログラム例】

```

DIM a(20)
FOR i = 1 TO 20
    READ a(i)
    IF a(i) = 0 THEN EXIT FOR
NEXT i
FOR i = 1 TO 20
    IF a(i) = 0 THEN EXIT FOR
    PRINT a(i);
NEXT i
END
DATA 5, 2, 3, 7, 8, 1, 2, 3, 9, 1
DATA 3, 8, 1, 7, 1, 0, 5, 2, 9, 9
    
```

データの読み込み
20 個分読む準備をする
データの読み込み
データが 0 のときループ終了
20 個分表示する準備をする
データが 0 のとき終了
データの表示

【結 果】

5 2 3 7 8 1 2 3 9 1 3 8 1 7 1

(39) 【プログラム例】

```

DO UNTIL x < 0
    INPUT x
    PRINT x
LOOP
END
    
```

x がマイナスになるまでくり返し
x を入力
x の表示

【結 果】

? 150 150, 0, 25, -54 を入力の例
150
? 0
0
? 25
25
? -54
-54

(40) 【プログラム例】

```

DO
    INPUT x
    z = x * x
    PRINT x; "^2"; "="; z
LOOP WHILE z < 10000
END
    
```

z が 10000 以下のときくり返し
x を入力
z に x^2 を代入
 x^2 を表示

【結 果】

? 55 55, 96, 150 を入力の例
55 ^2= 3025
? 96
96 ^2= 9216
? 150
150 ^2= 22500

(41) 【プログラム例】

```

t = 1000 ----- t の初期値
DO ----- t が 0 以下になるまでくり返し
  READ x ----- x にデータを読み込む
  t = t - x ----- t から x を順にひいていく
LOOP UNTIL t <= 0 -----
PRINT t ----- 結果の表示
END
DATA 120, 52, 456, 85, 92, 140, 358

```

【結 果】

-303

(42) 【プログラム例】

```

t = z = 0 ----- t と z の初期値, t は 1, 2, 3, ...をつくる, z は 1+2+3...の和
DO ----- z が 10000 以上になるまでくり返し
  t = t + 1 ----- t を 1, 2, 3...とする
  z = z + t ----- z に 1, 2, 3...を求める
LOOP UNTIL z >= 10000 -----
PRINT t; z ----- 結果の表示
END

```

【結 果】

141 10011

6章 条件分岐

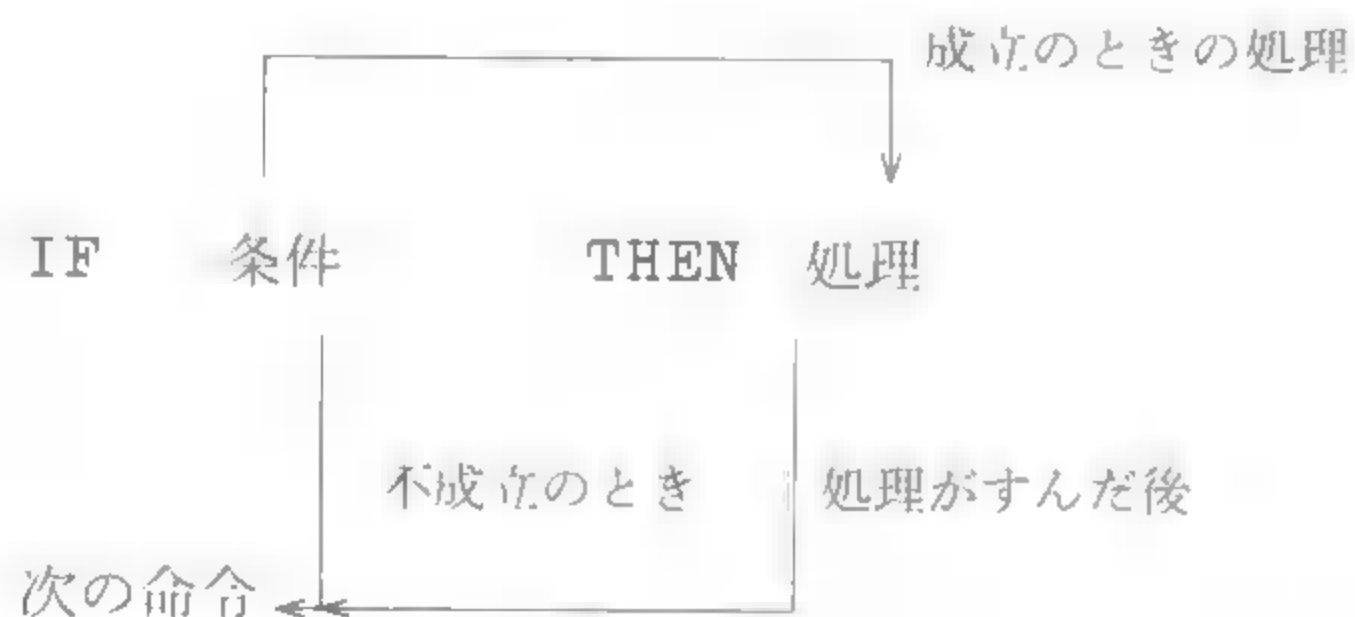
IF～THEN, IF～THEN～ELSE, ブロック IF,
END IF, ELSEIF, OR, AND, XOR, EQV, IMP, NOT

【例題 39】 IF～THEN

文字を入力し, "A" または先頭が "A" の文字列のとき表示するプログラムをつくれ.

【解 説】 ◎ IF～THEN～の分岐をします.

- ① IF 条件 THEN 処理によって条件が成立すると処理を行い条件が不成立のときは次へ進みます.



- ② 本例では入力された文字のASCIIコードが 65 すなわち "A" のとき, その文字を表示します. 65 でないときと, 表示がすむと終了です.

【プログラム例】

```

INPUT x$ ————— 文字の入力
IF ASC(x$) = 65 THEN PRINT x$ — 先頭が "A" のとき表示
END
  
```

【結 果】

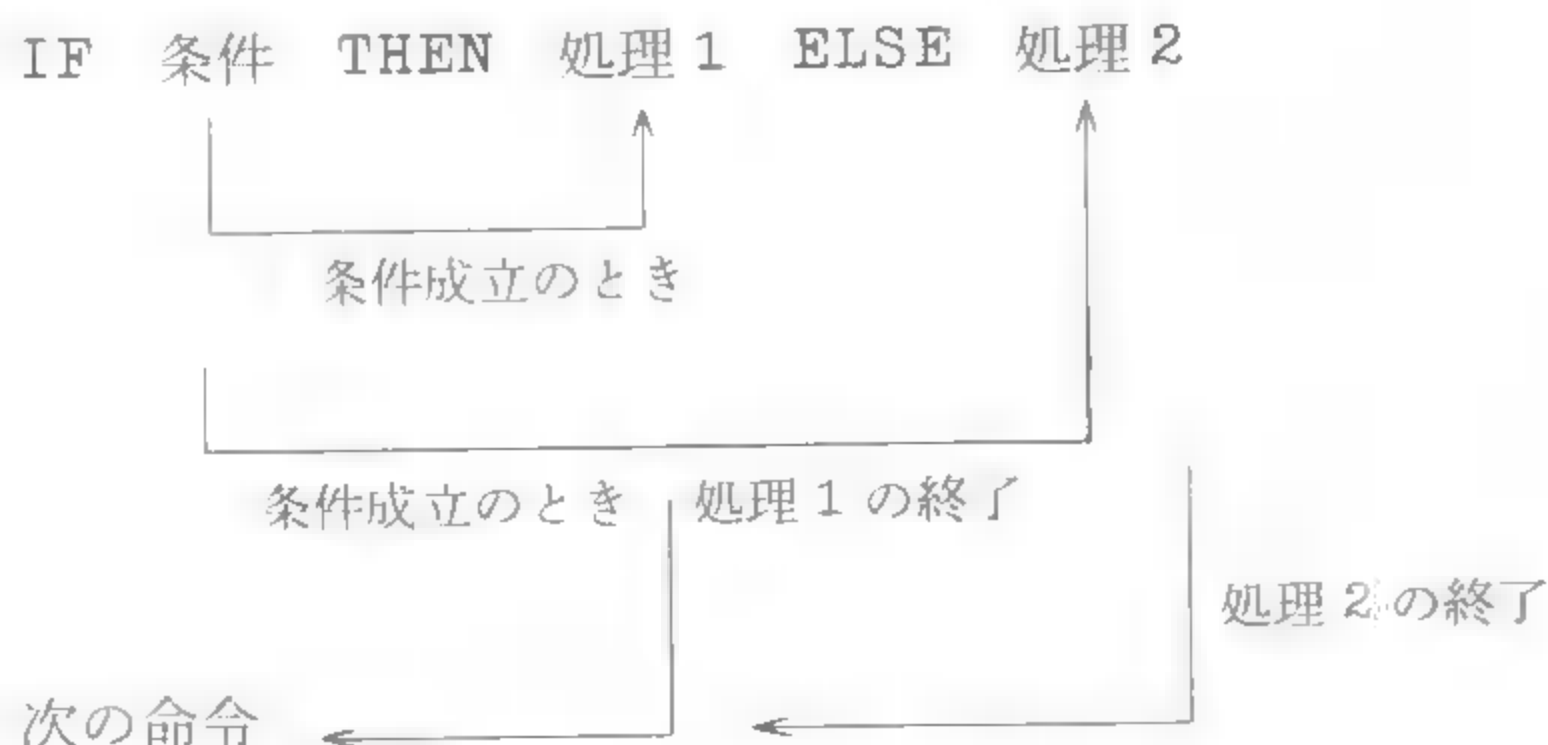
? Tokyo	Tokyo を入力の場合
? Akasaka	Akasaka を入力の場合
Akasaka	

【例題 40】 IF～THEN～ELSE～

数を入力し, 100 以上のとき ">=100" と表示し, それ以外では "<100" と表示するプログラムをつくれ.

【解 説】 ◎ IF 条件 THEN～ELSE～の分岐をします.

- ① 次の形で条件によって分岐します。



- ② 本例では x に値を入力し、100 以上のときは " ≥ 100 " と表示し、100 未満のときは " < 100 " と表示します。どちらも処理が終わると次の命令 END で終了します。

【プログラム例】

```

INPUT x ----- 数値の入力
IF x >= 100 THEN PRINT ">=100" ELSE PRINT "<100" ----- 2 条件に分けて表示
END

```

【結 果】

```

? 1250 ----- 1250 を入力の場合
>=100

? 10 ----- 10 を入力の場合
<100

```

【例題 41】 ブロック IF~THEN~ELSE

数 x を入力し、正のときは y に x を代入し、負のときは y に $-x$ を代入して \sqrt{y} を求めて表示するプログラムをつくれ。

【解 説】 ◎ ブロック IF~THEN~ELSE による分岐をします。

- ① ブロック IF は次の形で分岐をします。

IF 条件 THEN

処理 1 (命令群)

…条件が成立したとき行う処理
複数の命令をかくことができる

ELSE

処理 2 (命令群)

…条件が不成立のとき行う処理
複数の命令をかくことができる

END IF

処理 1、処理 2 の内容が複雑な場合、簡潔に表現できます。

- ② 処理 1 の内容が命令 1、命令 2、…命令 5、処理 2 の内容が命令 6、命令 7、…命令 9 でできているとします。ブロック IF を使うと次のようにつづきます。従来の BASIC と比較してください。

ブロック IF		従来の BASIC 例①	
IF	条件 THEN	IF 条件 THEN	命令 1 : 命令 2 : 命令 3 : 命令 4 :
			命令 5 ELSE 命令 6 : 命令 7 :
			命令 8 : 命令 9
命令 1		次の命令	
命令 2		従来の BASIC 例② (×××, ○○○, △△△は行番号)	
命令 3		IF 条件 THEN ××× ELSE ○○○	—— 一度×××と
			○○○に分けて処理先を示して分岐させる
命令 4	×××	命令 1	
命令 5		命令 2	
ELSE		命令 3	
命令 6		命令 4	
命令 7		命令 5	
命令 8		GOTO △△△	—— 処理がすむと GOTO 文で次の
命令 9	○○○	命令 6	命令の行へとばす
END IF		命令 7	
次の命令		命令 8	
		命令 9	
	△△△	次の命令	

- ③ 本例では処理 1, 処理 2 の内容が各々命令 1 つですからブロック IF を使わなくても簡単にかけます。ブロック IF があるからといってすべてブロック IF を使う必要はありません。

IF $x \geq 0$ THEN $y = x$ ELSE $y = -x$ (ブロック IF を使わない例)

【プログラム例】

```

INPUT x ————— x の入力
IF x >= 0 THEN ————— x が正のとき
    y = x —————
ELSE ————— x が負のとき
    y = -x —————
END IF
y = SQR(y) —————  $\sqrt{y}$  の表示
PRINT y —————
END

```

【結 果】

```

? 135          135 を入力の例
11.61895
? -547         -547 を入力の例
23.38803

```

【例題 42】 ブロック IF~THEN

次のデータを使い最も文字数の長い都市名を表示するプログラムをつくれ。

Chicago, Paris, Rome, London
Boston, Tokyo, Pittsburgh, Bon

【解 説】 ◎ ブロック IF~THEN を使って最大値を求めます。

- ① ブロック IF 条件 THEN 処理 END IF を使います。ELSE はありません。
- ② 最大値を求める方法は、前から順に都市名を読みその文字数と max を比較します。最初に max = 0 としておき、文字数が max より大きければその値を次の max とします。順にこれをくり返すと max に最大値が入ります。
- ③ 具体的にみてみましょう。

	1 回目	2 回目	3 回目
max 値 (最初は 0)	0	7	7
x\$	Chicago	Paris	Rome
文字数	7	5	4
max	7	7	7
y\$	Chicago	Chicago	Chicago

 7 回目	8 回目
	7	10
	Pittsburgh	Bon
	10	3
	10	10
	Pittsburgh	Pittsburgh

④ IF $l \geq \text{max}$ THEN

max = l

y\$ = x\$

END IF

の部分がブロック IF 文です。 $l \geq \text{max}$ のとき新しい l を max に入れ、新しい文字列 x\$ を y\$ に代入します。条件が不成立のとき、(新しい都市名がその前の最長文字数の都市名より短いとき) は特に処理はしませんので ELSE はありません。

【プログラム例】

```

max = 0
FOR i = 1 TO 8
  READ x$
  l = LEN(x$)
  IF l >= max THEN
    max = l
    y$ = x$
  END IF
NEXT i
PRINT y$
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

max = 0 ————— 最大値が入る
 FOR i = 1 TO 8 ————— 8 文字列の比較をくり返す
 READ x\$ ————— 文字列の読み込み
 l = LEN(x\$) ————— 文字数
 IF l >= max THEN ————— 文字数が大きいとき l を max, y\$ を x\$ とする
 max = l
 y\$ = x\$
 END IF
 NEXT i —————
 PRINT y\$ ————— y\$ (最長文字列) の表示
 END
 DATA Chicago, Paris, Rome, London
 DATA Boston, Tokyo, Pittsburgh, Bon

【結 果】

Pittsburgh

【例題 43】 ブロック IF の多重化

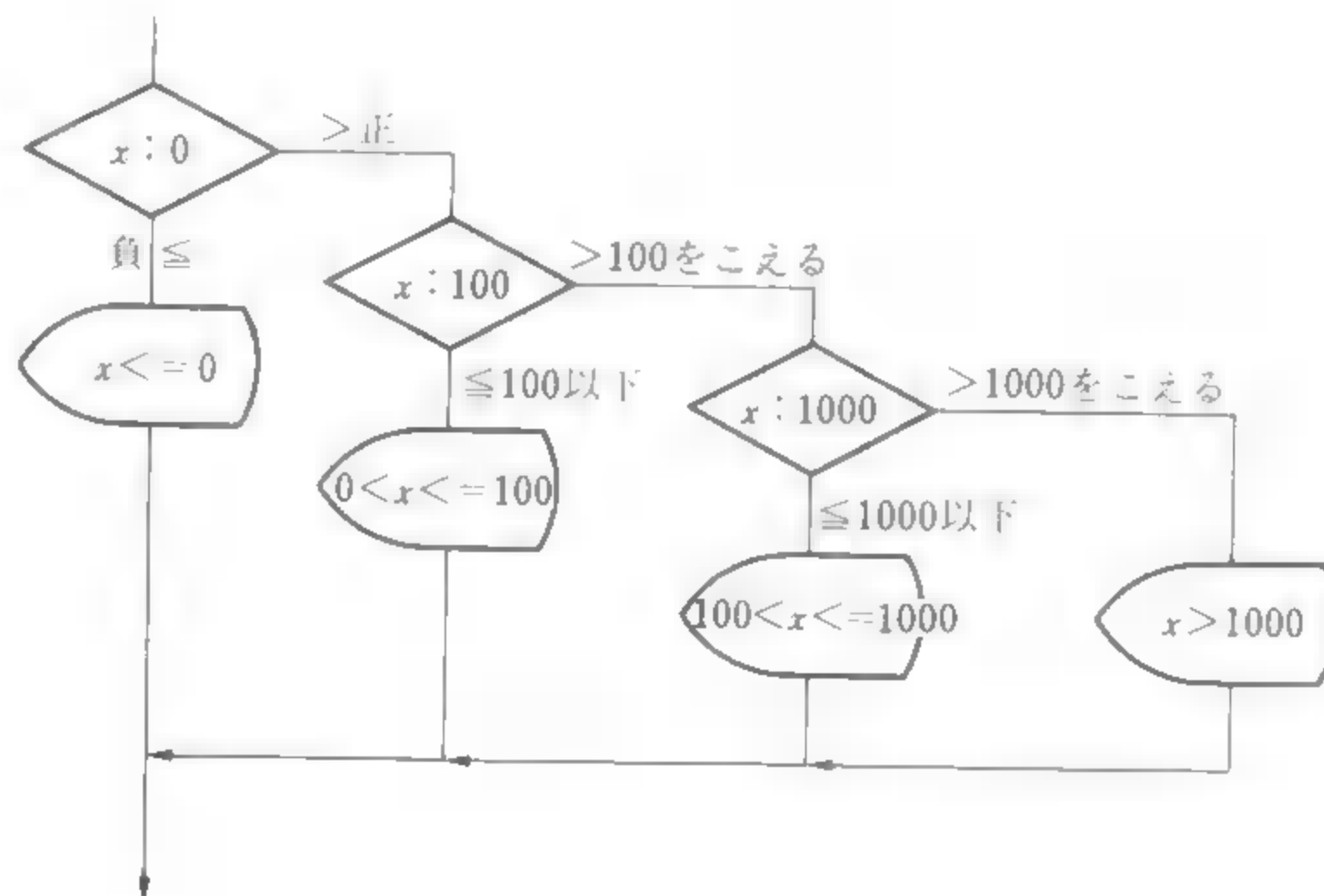
数 x を入力して負のとき " $x \leq 0$ ", 100 以下のとき " $0 < x \leq 100$ ", 1000 以下のとき " $100 < x \leq 1000$ ", 1000 をこえるとき " $x > 1000$ " と表示するプログラムをつくれ。

【解 説】 ㉟ ブロック IF の多重化をします。

① IF 文の中に IF 文を使うことができます。これを多重化といいます。IF 文は IF～THEN～ELSE と THEN と ELSE を持つもの、IF～THEN と THEN のみのものがありますので、あまり複雑にする とどの IF と THEN、ELSE が対応するのかわからなくなってしまいます。1 つのヒントは対応する THEN と ELSE の字下げの位置を同じにすることが考えられます。

② 本例では次のように考えます。

まず、数値が負か正かに分けます。負の場合は " $x \leq 0$ " と表示します。正の場合は 100 以下か、100 をこえるかに分けます。100 以下のときは " $0 < x \leq 100$ " と表示します。100 をこえるときは 1000 以下か 1000 をこえるかに分けます。1000 以下のときは " $100 < x \leq 1000$ " と表示し、1000 をこえるときは " $x > 1000$ " と表示します。



③ ELSE の中で IF 文を使う場合は ELSEIF を使います。

IF 条件 THEN

処理 1

条件 1 が成立するとき行う処理

条件 1 が不成立のときは処理 2 か処理 3 へ進む

ELSEIF 条件 2 THEN

処理 2

条件 2 が成立したときの処理

ELSE

処理 3

条件 2 が不成立のときの処理

END IF

【プログラム例】

```

INPUT x ----- x の入力
IF x <= 0 THEN ----- x が負のときの表示
    PRINT "x<=0"
ELSEIF x <= 100 THEN ----- x が正で 100 以下のときの表示
    PRINT "0<x<=100"
ELSEIF x <= 1000 THEN ----- x が 100 をこえ 1000 以下のときの表示
    PRINT "100<x<=1000"
ELSE ----- x が 1000 をこえるときに表示
    PRINT "x>1000";
END IF
END

```

【結 果】

```

? 358          358 を入力の例
100<x<=1000

? 3590         3590 を入力の例
x>1000

? -500         -500 を入力の例
x<=0

```

【例題 44】 OR 条件

下のデータのうち "R" で始まるか, "T" で始まる都市名を表示するプログラムをつくれ.
 データ;Chicago, Paris, Rome, London, Boston, Tokyo,
 Pittsburgh, Bon

【解 説】 ㉠ IF 条件の条件に OR を使って分岐します.

① IF 文の条件には関係演算子と組合わせとして下の論理演算子が使えます.

関係演算子の種類		論理演算子	
=	等しい	NOT	否定
>	大きい	AND	論理積
<	小さい	OR	論理和
<>	等しくない	XOR	排他的論理和
<=	以下	EQV	同値
>=	以上	IMP	包含

② 今回は OR を使います. OR はどちらかの条件が成立すれば全体の条件が成立します.

```
IF 条件 1 OR 条件 2 THEN 処理 1 ELSE 処理 2
```

条件 1 または条件 2 が成立すれば処理 1 を, どちらも不成立なら処理 2 を行います.

③ OR 条件は次のようになります.

X	Y	X OR Y
T	T	T
T	F	T
F	T	T
F	F	F

TはTRUE
FはFALSE

【プログラム例】

```

FOR i = 1 TO 8
  READ x$
  IF ASC(x$) = 82 OR ASC(x$) = 84 THEN PRINT x$
NEXT i
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon
  
```

【結 果】

Rome
Tokyo

【例題 45】 AND 条件

下のデータのうち 6 文字以下で最後に "n" を持つ文字列を表示するプログラムをつくれ。
データ;Chicago, Paris, Rome, London, Boston, Tokyo,
Pittsburgh, Bon

【解 説】 ◎ AND 条件を使って分岐します。

① IF 条件の中に AND を使うと 2 つの条件が同時に成立したとき分岐します。

IF LEN(x\$) <= 6 AND RIGHT\$(x\$) = "n" THEN PRINT x\$

条件 1 同時 条件 2
(6 文字以下) 成立 (最後が "n")

② AND 条件は次のようになります。

X	Y	X AND Y
T	T	T
T	F	F
F	T	F
F	F	F

TはTRUE
FはFALSE

【プログラム例】

```

FOR i = 1 TO 8
  READ x$
  IF LEN(x$) <= 6 AND RIGHT$(x$, 1) = "n" THEN PRINT x$
NEXT i
  
```

文字数が 6 以下で
最後が "n" のとき表示

```

END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

【結 果】

```

London
Boston
Bon

```

【例題 46】 NOT 条件

下のデータの中で文字数が 7 以下でない文字列を表示するプログラムをつくれ。

データ; Chicago, Paris, Rome, London, Boston, Tokyo, Pittsburgh, Bon

【解 説】 ◎ NOT 条件を使います。

- ① IF の条件の中に NOT を使うとその条件ではない場合分岐します。

```
IF NOT (LEN(x$) <= 7) THEN PRINT x$
```

文字数が 7 以下ではない

否定

```
IF LEN(x$) > 7 THEN PRINT x$と同じ
```

- ② NOT 条件は次のようになります。

X	NOT X
T	F
F	T

TはTRUE
FはFALSE

【プログラム例】

```

FOR i = 1 TO 8
  READ x$
  IF NOT (LEN(x$) <= 7) THEN PRINT x$ —— 文字数が 7 以下でないとき表示
NEXT i
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

【結 果】

```

Pittsburgh

```

【例題 47】 XOR 条件

次のデータのうち 4 文字以下か、最後に "n" を持つ 2 条件のどちらか 1 つが成立し、他が成立しない文字列を表示するプログラムをつくれ。

データ; Chicago, Paris, Rome, London, Boston, Tokyo, Pittsburgh, Bon

【解 説】 ◎ XOR 条件を使って分岐します。

- ① XOR は条件 1 XOR 条件 2 で一方の条件が成立し、他方が成立しない場合に真です。したがって、本例では文字数が 4 以下で最後が "n" でない場合か、文字数が 4 をこえて最後が "n" の場合です。

Chicago は文字数は 4 以下ではありませんし最後が "n" ではありません。条件 1, 2 ともに不成立ですから偽です。Rome は文字数が 4 文字以下で、最後が "n" ではありませんから真です。London は文字数が 4 以下ではありませんが最後が "n" ですから真です。Bon は文字数は 4 以下ですが、最後が "n" ですから条件 1, 2 がともに成立するため偽です。

- ② XOR 条件は次のようになります。

X	Y	X XOR Y
T	T	F
T	F	T
F	T	T
F	F	F

T は TRUE
F は FALSE

【プログラム例】

```
FOR i = 1 TO 8
  READ x$
  IF LEN(x$) <= 4 XOR RIGHT$(x$, 1) = "n" THEN PRINT x$
NEXT i
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon
```

—— 文字数が 4 以下で最後が "n" 以外かまたは文字数が 4 をこえ最後が "n" のとき表示

【結 果】

Rome
London
Boston

【例題 48】 EQV 条件

次のデータを使い、文字数が 4 以下で最後に "n" を持つものか、文字数が 4 以下ではなく、最後に "n" を持たないものを表示するプログラムをつくれ。

データ | Chicago, Paris, Rome, London, Boston, Tokyo, Pittsburgh, Bon

【解 説】 ◎ EQV 条件で分岐します。

- ① EQV は条件 1, 2 がともに成立するかともに不成立のとき真です。したがって、本例では文字数が 4 以下で最後に "n" を持つか、文字数が 4 をこえ最後に "n" を持たないとき真です。両方成立する文字列は Bon, 両方不成立は Chicago, Paris, Tokyo, Pittsburgh です。この両ケースが真です。片側が成立し、片側が不成立は Rome, London, Boston です。これは偽となります。
- ② EQV 条件は次のようになります。

X	Y	X EQV Y
T	T	T
T	F	F
F	T	F
F	F	T

TはTRUE
FはFALSE

【プログラム例】

```

FOR i = 1 TO 8
  READ x$
  IF LEN(x$) <= 4 EQV RIGHT$(x$, 1) = "n" THEN PRINT x$
NEXT i
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

—— 文字数が4以下で最後が"n"かまたは文字数が4をこえ最後が"n"ではないとき表示

【結 果】

```

Chicago
Paris
Tokyo
Pittsburgh
Bon

```

【例題 49】 IMP 条件

次のデータを使い、文字数が4以下で最後に"n"を持たないもの以外を表示するプログラムをつくれ。

データ; Chicago, Paris, Rome, London, Boston, Tokyo, Pittsburgh, Bon

【解 説】 ◎ IMP 条件で分岐します。

- ① IMP は条件1が真で条件2が偽のときのみ偽で、他はすべて真です。本例では Rome だけが偽です。
- ② IMP 条件は次のようになります。

X	Y	X IMP Y
T	T	T
T	F	F
F	T	T
F	F	T

TはTRUE
FはFALSE

【プログラム例】

```

FOR i = 1 TO 8
  READ x$

```

```

      IF LEN(x$) <= 4 IMP RIGHTS(x$, 1) = "n" THEN PRINT x$ —— 文字数が4以下で最後が"n"
NEXT i                                         でないもの以外を表示
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

【結 果】

```

Chicago
Paris
London
Boston
Tokyo
Pittsburgh
Bon

```

【例題 50】 論理演算子の組合わせ

次のデータを使い、文字数が5以下で最後が"n"のものか、最後が"o"のものを表示するプログラムをつくれ。

データ; Chicago, Paris, Rome, London, Boston, Tokyo, Pittsburgh, Bon

【解 説】 ◎ 論理演算子を組合わせて条件判断し分岐します。

- ① 第1の条件は文字数が5以下かつ最後が"n"のものです。この条件にあうものは Bon のみです。
- ② 第2の条件は最後が"o"です。この条件にあうものは Chicago, Tokyo です。
- ③ 条件1と条件2の OR は Bon, Chicago, Tokyo となります。

【プログラム例】

```

FOR i = 1 TO 8
  READ x$
  L = LEN(x$)
  p$ = RIGHTS(x$, 1)
  IF (L <= 5 AND p$ = "n") OR (p$ = "o") THEN PRINT x$ —— 文字数が5以下で最後が"n"
NEXT i                                         かまたは最後が"o"のものを
END                                           表示
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

【結 果】

```

Chicago
Tokyo
Bon

```

〔演習問題〕

- (43) 下のデータを読み"P"で始まる都市名を表示するプログラムをつくれ。

データ; Chicago, Paris, Rome, London, Boston, Tokyo, Pittsburgh, Bon

- (44) (43)のデータを使い最後が"n"で終わる都市名を表示するプログラムをつくれ。

- (45) (43) のデータを使い 5 文字でかつ "P" で始まる都市名を表示するプログラムをつくれ。
- (46) (43) のデータを使い 6 文字以上のとき都市名と文字数を、それ以外では都市名のみを表示するプログラムをつくれ。
- (47) アルファベット A~Z, a~z を入力し、H とそれより後の文字のときはそのまま表示し、それより前のときは小文字にして表示するプログラムをつくれ。
- (48) x を入力して正のとき y を入力し $x*y$ を表示し、 x が負のときは x^3 を表示するプログラムをつくれ。
- (49) x を入力して、 x が正のとき a を 2, b を 5, c を 3 とし、 x が負のとき a を 4, b を 2, c を 7 とし $y=ax^2+bx+c$ を求めて表示するプログラムをつくれ。
- (50) (43) のデータを使い文字数が最も少ない都市名を表示するプログラムをつくれ。
- (51) (43) のデータを使い "B" または "P" で始まる都市名を表示するプログラムをつくれ。
- (52) 文字列を入力し、下のデータの中にあれば表示し、なければ "END" と表示するプログラムをつくれ。

データ;Tokyo, Osaka, Nagoya, Fukuoka, Sapporo, Okayama

- (53) (43) のデータを使って、文字数が 4 未満か 8 をこえるものを表示するプログラムをつくれ。
- (54) (43) のデータを使って、文字数が 4 以下か 8 をこえるものでかつ最後が "n" のものを表示するプログラムをつくれ。
- (55) (43) のデータを使って、文字数が 6 をこえて最後が "n" のものか、文字数が 6 をこえ、最後が "n" でないものを表示するプログラムをつくれ。
- (56) (43) のデータを使って、文字数が 6 をこえて Chicago ではないものを表示するプログラムをつくれ。

〔解 答〕

(43) 【プログラム例】

```

FOR i = 1 TO 8 ————— データの読み込みのくり返し
  READ x$ ————— x$にデータを読み込む
  IF ASC(x$) = 80 THEN PRINT x$ ——"P"で始まるとき x$を表示
NEXT i —————
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

【結 果】

Paris
Pittsburgh

(44) 【プログラム例】

```

FOR i = 1 TO 8 ————— データの読み込みのくり返し
  READ x$ ————— x$にデータを読み込む
  IF RIGHT$(x$, 1) = "n" THEN PRINT x$ ————— 右端がnのとき表示
NEXT i —————
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

【結 果】

```

London
Boston
Bon

```

(45) 【プログラム例】

```

FOR i = 1 TO 8 ————— データの読み込みのくり返し
  READ x$ ————— x$にデータを読み込む
  l = LEN(x$) ————— x$の長さをlに代入
  IF l = 5 AND ASC(x$) = 80 THEN PRINT x$ ————— lが5で先頭文字が"P"のとき表示
NEXT i —————
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

【結 果】

```

Paris

```

(46) 【プログラム例】

```

FOR i = 1 TO 8 ————— 8回のくり返し
  READ x$ ————— x$にデータを読む
  IF LEN(x$) >= 6 THEN PRINT x$; LEN(x$) ELSE PRINT x$ ————— 6文字以上のとき都市名と文字数を
NEXT i ————— 表示, それ以外は都市名のみ表示
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

【結 果】

```

Chicago 7
Paris
Rome
London 6
Boston 6
Tokyo
Pittsburgh 10
Bon

```

(47) 【プログラム例】

```

INPUT x$ ————— x$の入力
IF x$ >= "H" THEN PRINT x$ ELSE PRINT CHR$(ASC(x$) + 32) ————— "H"とそれより後のとき表示, それ
END ————— 以外は小文字にして表示

```

【結 果】

? A A を入力の場合
a

? P P を入力の場合
P

? d d を入力の場合
d

(48) 【プログラム例】

```

INPUT x ----- x の入力
IF x >= 0 THEN ----- x が正のとき
    INPUT y ----- y を入力して, x*y を表示
    PRINT x * y -----
ELSE ----- x が負のとき
    PRINT x * x * x ----- x^3 を表示
END IF
END
    
```

【結 果】

? 6 x に 6, y=7 を入力の場合
? 7
42

? -88 x に -88 を入力の場合
-681472

(49) 【プログラム例】

```

INPUT x ----- x の入力
IF x >= 0 THEN ----- x が正のとき
    a = 2: b = 5: c = 3 ----- a=2, b=5, c=3 とする
ELSE ----- x が負のとき
    a = 4: b = 2: c = 7 ----- a=4, b=2, c=7 とする
END IF
y = a * x * x + b * x + c ----- y = ax^2 + bx + c を求める
PRINT a; "x^2+"; b; "x+"; c; "="; y ----- 結果の表示
END
    
```

【結 果】

? 8 8 を入力の場合
2 x^2+ 5 x+ 3 = 171

? -3 -3 を入力の場合
4 x^2+ 2 x+ 7 = 37

(50) 【プログラム例】

```

min = 20
FOR i = 1 TO 8
  READ x$
  l = LEN(x$)
  IF l <= min THEN
    min = l
    y$ = x$
  END IF
NEXT i
PRINT y$
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

最小の文字の都市名を求める

文字数がそのときの最小値 m より小さければ
その文字数を m に代入, そのときの都市名を
 $y\$$ に代入

【結果】

Bon

(51) 【プログラム例】

```

FOR i = 1 TO 8
  READ x$
  IF ASC(x$) = 66 THEN
    PRINT x$
  ELSEIF ASC(x$) = 80 THEN
    PRINT x$
  END IF
NEXT i
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

データの読み込みのくり返し

$x\$$ にデータを読み込む

"B" で始まる时表示

"P" で始まる时表示

【結果】

Paris
Boston
Pittsburgh
Bon

(52) 【プログラム例】

```

INPUT x$
DO UNTIL x$ = y$
  READ y$
  IF y$ = "End" THEN x$ = y$
LOOP
PRINT x$
END
DATA Tokyo, Osaka, Nagoya
DATA Fukuoka, Sapporo, Okayama
DATA End

```

$x\$$ の入力

$x\$ = y\$$ で終了, 同じデータがあればループの外へ出る

データを読む

データを読んで "END" になれば $x\$$ を "END" にしてループの外へ出る

$x\$$ の表示

【結 果】

? Sapporo Sapporo を入力の場合
Sapporo

? Kyoto Kyoto を入力の場合
End

(53) 【プログラム例】

```
FOR i = 1 TO 8
  READ x$
  l = LEN(x$)
  IF (l < 4) OR (l > 8) THEN PRINT x$——文字数が4未満か8をこえるものを表示
NEXT i
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon
```

【結 果】

Pittsburgh
Bon

(54) 【プログラム例】

```
FOR i = 1 TO 8
  READ x$
  l = LEN(x$)
  p$ = RIGHT$(x$, 1)
  IF (l <= 4 OR l > 8) AND (p$ = "n") THEN PRINT x$——文字数が4以下か8をこえ、かつ最後が"n"のとき表示
NEXT i
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon
```

【結 果】

Bon

(55) 【プログラム例】

```
FOR i = 1 TO 8
  READ x$
  IF LEN(x$) > 6 EQV RIGHT$(x$, 1) = "n" THEN PRINT x$——文字数が6をこえ最後が"n"または文字数が6以下で最後が"n"でないとき表示
NEXT i
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon
```

【結 果】

Paris
Rome
Tokyo

(56) 【プログラム例】

```

FOR i = 1 TO 8
  READ x$
  IF LEN(x$) > 6 AND NOT (x$ = "Chicago") THEN PRINT x$
NEXT i
END
DATA Chicago, Paris, Rome, London
DATA Boston, Tokyo, Pittsburgh, Bon

```

—— 文字数が6をこえかつ
Chicago でないとき
表示

【結 果】

Pittsburgh

7章 多分岐

SELECT～CASE, END SELECT

【例題 51】 SELECT～CASE による多分岐

数値を入力し、1 のとき "A"、2 のとき "B"、3 のとき "C"、4 のとき "D"、5 のとき "E" を表示し、それ以外では "END" と表示するプログラムをつくれ。

【解 説】 ◎ SELECT～CASE による多分岐をします。

- ① 条件によって分岐先を 3 つ以上にしたい場合 SELECT～CASE を使います。
- ② 本例は数値を入力し、1、2、3、4、5 の場合とそれ以外の場合の 6 つの分岐をします。その形は次のとおりです。

SELECT x%	_____	x%の値によって分岐先が決まる
CASE 1	_____	x%が 1 のとき
PRINT "A"	_____	x%が 1 のときの処理。複数の文を書いてよい。終了すると END SELECT の次へ進む
CASE 2	_____	x%が 2 のとき
PRINT "B"	_____	
CASE 3	_____	x%が 3 のとき
PRINT "C"	_____	
CASE 4	_____	x%が 4 のとき
PRINT "D"	_____	
CASE 5	_____	x%が 5 のとき
PRINT "E"	_____	
CASE ELSE	_____	1、2、3、4、5 以外の場合
PRINT "END"	_____	
END SELECT	_____	SELECT 文の終了

- ③ x%が 1 のときは CASE1 へ進み "A" を表示して END SELECT の次へ進みます。PRINT "A" の部分が 1 の場合の処理内容です。ここは複数の文をかくことができます。x%が 2、3、4、5 の場合は各々 CASE2、CASE3、CASE4、CASE5 へ分岐し、各々の処理をします。
- ④ x%が 1、2、3、4、5 以外では CASE ELSE へ分岐し、その処理をします。x%が 1、2、3、4、5 しかなければ CASE ELSE はいりませんが、x%が 1～5 以外があつて、CASE ELSE がないとエラーとなります。

- ⑤ 従来の BASIC では ON～GOSUB または ON～GOTO でかくことができますが、かなりみにくいものとなります。

ON～GOSUB を使った例

(○○などは行番号を示す)

```
ON x% GOSUB  ○○,  ××,  △△,  □□,  ●●,  ▲▲
)

○○ PRINT "A"
  RETURN
×× PRINT "B"
  RETURN
△△ PRINT "C"
  RETURN
□□ PRINT "D"
  RETURN
●● PRINT "E"
  RETURN
▲▲ PRINT "END"
  RETURN
```

ON～GOTO を使った例

(○○などは行番号を示す)

```
ON x% GOTO  ○○,  ××,  △△,  □□,  ●●,  ▲▲
●● )

○○ PRINT "A"
  GOTO  ●●
×× PRINT "B"
  GOTO  ●●
△△ PRINT "C"
  GOTO  ●●
□□ PRINT "D"
  GOTO  ●●
●● PRINT "E"
  GOTO  ●●
▲▲ PRINT "END"
  GOTO  ●●
```

【プログラム例】

```

INPUT x% ----- 整数の入力
SELECT CASE x% ----- x%の値によって分岐
  CASE 1 ----- x%が1のとき
    PRINT "A"
  CASE 2 ----- x%が2のとき
    PRINT "B"
  CASE 3 ----- x%が3のとき
    PRINT "C"
  CASE 4 ----- x%が4のとき
    PRINT "D"
  CASE 5 ----- x%が5のとき
    PRINT "E"
  CASE ELSE ----- x%が1~5以外の場合
    PRINT "END"
END SELECT ----- 分岐の終了
END

```

【結 果】

```

? 3          3を入力の場合
C

? 5          5を入力の場合
E

? 15         15を入力の場合
END

```

【例題 52】 文字による多分岐をします

文字を入力し、"A"のとき"Apple"、"B"のとき"Bench"、"C"のとき"Chair"、"D"のとき"Desk"、"E"のとき"Election"、それ以外では"END"と表示するプログラムをつくれ。

【解 説】 ◎ SELECT 文字 CASE によって多分岐をします。

- ① 文字を条件として多分岐をします。この場合は CASE "A"のように文字を"でかこみます。あとは数値の場合と同じです。

【プログラム例】

```

INPUT x$ ----- x$の入力
SELECT CASE x$ ----- x$の値によって分岐
  CASE "A" ----- x$が"A"のとき
    PRINT "Apple"
  CASE "B" ----- x$が"B"のとき
    PRINT "Bench"
  CASE "C" ----- x$が"C"のとき
    PRINT "Chair"
  CASE "D" ----- x$が"D"のとき
    PRINT "Desk"

```

```

CASE "E"——— x$が"E"のとき
  PRINT "Election"
CASE ELSE——— x$がA～E以外のとき
  PRINT "END"
END SELECT——— 分岐の終了
END

```

【結 果】

? B B を入力の場合
Bench

? a a を入力の場合
END

【例題 53】 範囲で分岐

数値を入力し、1000 以上のとき " $x \geq 1000$ ", 500 から 999 までのとき " $500 \leq x < 1000$ ", 300 から 499 のとき " $300 \leq x < 500$ ", 100 から 299 のとき " $100 \leq x < 300$ ", 0 から 99 までのとき " $0 \leq x < 100$ ", それ以外では "マイナス" と表示するプログラムをつくれ。ただし、上の表示のうち x は各々実際に入力された値を表示するものとする。たとえば、125 が入力されれば " $100 \leq 125 < 300$ " と表示するものとする。

【解 説】 ◎ CASE の後に範囲を指定します。

- ① CASE IS ≥ 1000 では値が 1000 以上の場合の分岐先です。IS ≥ 1000 が 1000 以上を示します。比較演算子として $>$, \geq , $<$, \leq , $<>$, $=$ が使えます。ただし、 $=$ はたとえば、CASE 5 のように単に値をかくだけですみます。
- ② CASE 500 TO 999 は 500 から 999 のときの分岐先です。値 1 TO 値 2 で値 1 から値 2 までの範囲を指定できます。ただし、必ず値 1 $<$ 値 2 である必要があります。

【プログラム例】

```

INPUT x%——— x%の入力
SELECT CASE x%——— x%の値によって分岐
  CASE IS  $\geq 1000$ ——— x%が1000以上のとき
    PRINT x%; ">=1000"
  CASE 500 TO 999——— x%が500～999のとき
    PRINT "500<="; x%; "<1000"
  CASE 300 TO 499——— x%が300～499のとき
    PRINT "300<="; x%; "<500"
  CASE 100 TO 299——— x%が100～299のとき
    PRINT "100<="; x%; "<300"
  CASE 0 TO 99——— x%が0～99のとき
    PRINT "0<="; x%; "<100"
  CASE ELSE——— x%が0未満のとき
    PRINT "マイナス"
END SELECT——— 分岐の終了
END

```

【結 果】

? 1563 1563 を入力の場合
1563 >=1000

? 48 48 を入力の場合
0<= 48 <100

? -1200 -1200 を入力の場合
マイナス

【例題 54】 CASE に複数の範囲を指定します

整数を入力し、負のときはその値を、1 から 10、20 から 30、40 から 50 のときは 2 乗の値、11 から 19、21 から 29、31 から 39 のときは平方根、50 をこえるときはその値を表示するプログラムをつくれ。

【解 説】 ◎ CASE に複数個の範囲を指定します。

- ① CASE 1 TO 10, 20 TO 30, 40 TO 50 によって 1 から 10、20 から 30、40 から 50 の 3 つの範囲を指定することができます。

【プログラム例】

```

INPUT x% ----- x%の入力
SELECT CASE x% ----- x%の値によって分岐
  CASE IS <= 0 ----- x%が 0 以下のとき
    PRINT x%
  CASE 1 TO 10, 20 TO 30, 40 TO 50 ----- x%が 1~10, 20~30, 40~50 のとき
    PRINT x% * x%
  CASE 11 TO 19, 21 TO 29, 31 TO 39 ----- x%が 11~19, 21~29, 31~39 のとき
    PRINT SQR(x%)
  CASE IS > 50 ----- x%が 50 より大きいとき
    PRINT x%
END SELECT ----- 分岐の終了
END

```

【結 果】

? 25 25 を入力の場合
625

? 15 15 を入力の場合
3.872983

? -85 -85 を入力の場合
-85

【例題 55】 多分岐でサブルーチン呼び出す

次のメニュー画面を表示し、1 のとき和、2 のとき差、3 のとき積、4 のとき商、それ以外では再入力するプログラムをつくれ。ただし、数値が入力されたら演算サブルーチンへ各々分岐して、数を2つ入力して演算し、結果を表示するものとする。

和 = 1
差 = 2
積 = 3
商 = 4

1 から 4

【解 説】 ◎ 多分岐とサブルーチンでメインルーチンをすっきりさせます。

- ① メインルーチンは入力された値 1～4 とそれ以外で各々多分岐するものです。【例題 51】と同じ構造です。
- ② 各々の case では call a1(a,b,c)～a4(a,b,c)でサブルーチン a1～a4 を呼び出します。
- ③ 結果はメインルーチンの最後で表示します。

【プログラム例】

```

DECLARE SUB a1 (x!, y!, z!)
DECLARE SUB a2 (x!, y!, z!)
DECLARE SUB a3 (x!, y!, z!)
DECLARE SUB a4 (x!, y!, z!)
PRINT "和   = 1"
PRINT "差   = 2"
PRINT "積   = 3"
PRINT "商   = 4"
PRINT
re: INPUT "1 から 4 "; x%
SELECT CASE x%
CASE 1
    d$ = "+"
    CALL a1(a, b, c)
CASE 2
    d$ = "-"
    CALL a2(a, b, c)
CASE 3
    d$ = "*"
    CALL a3(a, b, c)
CASE 4
    d$ = "/"
    CALL a4(a, b, c)
CASE ELSE
    GOTO re:
END SELECT
PRINT a; d$; b; "="; c
END

```

メニュー画面

x%の入力

x%の値で分岐

1 のとき

2 のとき

3 のとき

4 のとき

1～4 以外するとき再入力

結果の表示

```

SUB a1 (x, y, z) ----- サブルーチン a1
  INPUT x, y
  z = x + y
END SUB -----

```

```

SUB a2 (x, y, z) ----- サブルーチン a2
  INPUT x, y
  z = x - y
END SUB -----

```

```

SUB a3 (x, y, z) ----- サブルーチン a3
  INPUT x, y
  z = x * y
END SUB -----

```

```

SUB a4 (x, y, z) ----- サブルーチン a4
  INPUT x, y
  z = x / y
END SUB -----

```

【結 果】

和 = 1 3 を入力し, 5, 4 を入力した例
 差 = 2
 積 = 3
 商 = 4

1 から 4 ? 3
 ? 5, 4
 5 * 4 = 20

和 = 1 2 を入力し, 7, 3 を入力した例
 差 = 2
 積 = 3
 商 = 4

1 から 4 ? 2
 ? 7, 3
 7 - 3 = 4

和 = 1 4 を入力し, 99, 3 を入力した例
 差 = 2
 積 = 3
 商 = 4

1 から 4 ? 4
 ? 99, 3
 99 / 3 = 33

和 = 1 8, 7, 1 を入力し, 50, 24 を入力した例
 差 = 2
 積 = 3
 商 = 4

1 から 4 ? 8
 1 から 4 ? 7
 1 から 4 ? 1
 ? 50, 24
 50 + 24 = 74

〔演習問題〕

- (57) 数を入力し、1 のとき "first", 2 のとき "second", 3 のとき "third" と表示し、それ以外では入力された値を表示するプログラムをつくれ。
- (58) 文字を入力し、"A" のとき "ABC", "B" のとき "BCD", "C" のとき "CDE", それ以外 のとき "XYZ" と表示するプログラムをつくれ。
- (59) 整数を入力し、65 から 90 のときはその値をアスキーコードとする文字を、97 から 122 までのときはその値をアスキーコードとする文字を表示し、それ以外では "アルファベット イガイ" と表示するプログラムをつくれ。
- (60) 整数を入力して 1 のとき、サブルーチン aa を呼び出し "ニュウリヨクチ=1", 2 のとき、サブルーチン bb を呼び出し "ニュウリヨクチ=2", それ以外のときはサブルーチン cc と呼び出し "リュウリヨクチ ハ 1,2 イガイ" と表示するプログラムをつくれ。
- (61) 整数 x% を入力し、1~5 のとき x%! (階乗), 6~9 のとき 2x% (累乗), 10~15 のとき、1 から x% までの和を求めそれ以外は再入力するプログラムをつくれ。
- (62) 英文字列を入力し、"January" のときは "January ハ 31 ニチ", "Feburary" のとき "Ferbruary ハ 28 ニチ", ... "June" のとき "June ハ 30 ニチ" と表示し、その他のときは再入力するプログラムをつくれ。
- (63) アルファベットの大文字または小文字, 数字, またはカナを 1 つ入力し、小文字のとき "アルファベット コモジ", 大文字のとき "アルファベット オオモジ", 数字のとき "スウジ", カナのとき "カナ" と表示するプログラムをつくれ。

〔解 答〕

(57) 【プログラム例】

```

INPUT x% ----- x%の入力
SELECT CASE x%
  CASE 1 ----- 1 のとき
    PRINT " first"
  CASE 2 ----- 2 のとき
    PRINT "second"
  CASE 3 ----- 3 のとき
    PRINT "third"
  CASE ELSE ----- 1~3 以外のとき
    PRINT x%
END SELECT
END

```

【結 果】

? 1	1 を入力の例
first	
? 3	3 を入力の例
third	
? 120	120 を入力の例
120	

(58) 【プログラム例】

```

INPUT x$ ————— x$を入力
SELECT CASE x$
  CASE "A" ————— A のとき
    PRINT "ABC" —————
  CASE "B" ————— B のとき
    PRINT "BCD" —————
  CASE "C" ————— C のとき
    PRINT "CDE" —————
  CASE ELSE ————— A～C 以外するとき
    PRINT "XYZ" —————
END SELECT
END

```

【結 果】

```

? A      A を入力の場合
ABC
? f      f を入力の場合
XYZ

```

(59) 【プログラム例】

```

INPUT x% ————— x%の入力
SELECT CASE x%
  CASE 65 TO 90 ————— 65～90 のとき
    PRINT CHR$(x%) —————
  CASE 97 TO 122 ————— 97～122 のとき
    PRINT CHR$(x%) —————
  CASE ELSE ————— その他のとき
    PRINT "アルファベット イカイ" —————
END SELECT
END

```

【結 果】

```

? 69      69 を入力の場合
E
? 115     115 を入力の場合
S
? 500     500 を入力の場合
アルファベット イカイ

```

(60) 【プログラム例】

```

DECLARE SUB aa () ————— サブルーチンの宣言
DECLARE SUB bb ()
DECLARE SUB cc ()
INPUT x% ————— x%の入力
SELECT CASE x%
  CASE 1 ————— 1 のとき
    CALL aa —————
  CASE 2 ————— 2 のとき
    CALL bb —————
  CASE ELSE ————— 1～2 以外するとき
    CALL cc —————
END SELECT

```

END

```
SUB aa-----サブルーチン aa
PRINT "ニューリョクチ=1"
END SUB
```

```
SUB bb _____ サブルーチン bb
PRINT "ニュウリョクチ=2"
END SUB
```

```
SUB cc ----- サブルーチン cc
PRINT "ニュウリョクチ ハ 1,2 イガ イ"
END SUB
```

【結 果】

2 を人力の例

? 1 1 を入力の場合

ニユウリョクチ=1

? 453
 ニュウリョクチ ハ 1,2 イカ`イ 453 を入力 of 例

(61) 【プログラム例】

```

re: INPUT x% ----- x%の入力
SELECT CASE x% ----- x%の値で分岐
  CASE 1 TO 5 ----- 1～5 のとき
    z% = 1 ----- z%に階乗を求める
    FOR i% = x% TO 1 STEP -1
      z% = z% * i%
    NEXT i%
    PRINT z% -----
  CASE 6 TO 9 ----- 6 から 9 のとき
    PRINT 2 ^ x% ----- 2x% を表示
  CASE 10 TO 15 ----- 10 から 15 のとき
    w% = 0 ----- wa%に和を求める
    FOR i% = 1 TO x%
      wa% = wa% + i%
    NEXT i%
    PRINT wa% -----
  CASE ELSE ----- 1～15 以外のとき再入力
    GOTO re: -----
END SELECT
END

```

【結 果】

```

? 9          9 を入力の場合
512
? 4          4 を入力の場合
24
? 14         14 を入力の場合
105
? 85         85 を入力し、再入力で 1 を入力の場合
? 1
1

```

(62) 【プログラム例】

```

re: INPUT x$ ----- 文字列の入力
SELECT CASE x$ ----- 文字列 x$ によって分岐
  CASE "January" ----- "January" のとき
    PRINT "January は 31 日"
  CASE "February" ----- "February" のとき
    PRINT "February は 28 日"
  CASE "March" ----- "March" のとき
    PRINT "March は 31 日"
  CASE "April" ----- "April" のとき
    PRINT "April は 30 日"
  CASE "May" ----- "May" のとき
    PRINT "May は 31 日"
  CASE "June" ----- "June" のとき
    PRINT "June は 30 日"
  CASE ELSE ----- その他のとき再入力
    GOTO re:
END SELECT
END

```

【結 果】

```

? May          May を入力の場合
May は 31 日
? February     February を入力の場合
February は 28 日

```

(63) 【プログラム例】

```

re: INPUT x$ ----- 文字列 x$ の入力
xx% = ASC(x$) ----- x$ のアスキーコードを xx% に代入
SELECT CASE xx% ----- xx% の値で分岐
  CASE 48 TO 57 ----- 48~57 のとき ("0"~"9" のとき)
    PRINT "数字"
  CASE 65 TO 90 ----- 65~90 のとき ("A"~"Z" のとき)
    PRINT "アルファベット 大文字"
  CASE 97 TO 122 ----- 97~122 のとき ("a"~"z" のとき)
    PRINT "アルファベット 小文字"
  CASE 166 TO 221 ----- 166~221 のとき ("ア"~"ン" のとき)
    PRINT "かな"

```



```

CASE ELSE _____ その他のとき再入力
      GOTO re: _____
END SELECT
END

```

【結 果】

```

? 65 _____ 65 を入力の例
スウシゝ
? A _____ A を入力の例
アルファベット オオモシゝ
? x _____ x を入力の例
アルファベット コモシゝ
? シ _____ シを入力の例
カナ

```

8 章 配 列

DIM, OPTION BASE

【例題 56】 配 列

下のデータを配列 a%(0)~a%(9)に代入して、a%(5)と a%(7)を表示するプログラムをつくれ。

データ: 7, 5, 2, 9, 7, 3, 8, 4, 1, 0

【解 説】 ◎ 1 次数値配列を使います。

① DIM a%(10)で a%(0), a%(1), …a%(9)の 10 個の整数型配列変数を確保します。これは DIM a(10) AS INTEGER と宣言することもできます。

② FOR i=0 TO 9

READ a%(i)

NEXT i

によってデータを順に a%(0), a%(1), …a%(9)へ読み込みます。1 度データが配列変数に代入されればあとはどれでも任意にとり出せます。

③ PRINT a%(5);a%(7)によって a%(5)の 3 と a%(7)の 4 を表示します。

④ a%(10)の 10 を添字といい最大 32,768 まで使えます。整数型の場合 2 バイトを使用しますから a%(32,768)は約 65k バイト使用します。

【プログラム例】

```
DIM a%(10) —— 配列 a%(0)~a%(9)の宣言
FOR i = 0 TO 9 —— i を 0~9 までループ
  READ a%(i) —— a%(0)~a%(9)にデータを読み込む
NEXT i
PRINT a%(5); a%(7) —— a%(5)と a%(7)の表示
END
DATA 7, 5, 2, 9, 7, 3, 8, 4, 1, 0
```

【結 果】

3 4

【例題 57】 添字を任意に決定

添字を-5 から 5 までとする配列 c%()を宣言して、下のデータを順に代入し、c%(0)の値を表示するプログラムをつくれ。

データ: 3, 8, 7, 2, 1, 9, 6, 3, 5, 2, 7

【解 説】 ◎ 添字の範囲を任意に決めます。

- ① DIM c%(-5 TO 5)はc%(-5), c%(-4), c%(-3), …c%(0), c%(1), …c%(5)の11個の領域を確保します。

Quick BASICでは添字の下限と上限を自由に設定できるようになった他マイナスを使うこともできます。この便利さについては次の【例題58】で示します。

【プログラム例】

```

DIM c%(-5 TO 5)  ----- 配列 c%の宣言
FOR i = -5 TO 5  -----  c%(-5)～c%(5)にデータを読み込む
    READ c%(i)
NEXT i
PRINT c%(0); -----  c%(0)の表示
END
DATA 3, 8, 7, 2, 1, 9, 6, 3, 5, 2, 7

```

【結 果】

9

【例題58】 xを-3から+3まで $y=x^3+x^2+x$ を求める

x^3+x^2+x の値をxを-3から+3まで増分を1として求め配列y%()に格納して表示するプログラムをつくれ。

【解 説】 ◎ 添字にマイナス値を使って関数の値を求めます。

- ① x^3+x^2+x の値をxを-3から+3までについて求めます。このとき、結果を入れる配列をy%()とするとFOR x=-3 TO 3のくり返しでxを-3, -2, -1, 0, 1, 2, 3として、関数の値を求め、それをy%(x)に代入できます。添字としてxをそのまま使えますから非常に簡単です。
- ② 添字を0～6として求める従来のBASICでは次のようになります。

```

FOR x=-3 TO 3
    Y(x+3)=x^3+x^2+x      ← x+3によって添字を0, 1, 2, …6としている。
NEXT I

```

【プログラム例】

```

DIM y%(-3 TO 3) ----- 配列 y%(-3)～y%(3)の宣言
FOR x = -3 TO 3 -----  xを-3から3まで $x^3+x^2+x$ を求めてy%(-3)～y%(3)に代入
    y%(x) = x * x * x + x * x + x
NEXT x
FOR x = -3 TO 3 -----  y%(-3)～y%(3)の表示
    PRINT y%(x);
NEXT x
END

```

【結 果】

-21 -6 -1 0 3 14 39

【例題 59】 1 次元文字列配列とオプションベース

文字列配列 $b\$(1) \sim b\(5) に下のデータを読み、最初と最後を表示するプログラムをつくれ。
 データ: Yamada, Osaki, Ueda, Yoshino, Honda

【解 説】 ◎ 文字列配列とオプションベースを使います。

- ① OPTION BASE 1 によって、配列の添字を 1 からとります。したがって、 $\text{DIM } b\$(5)$ は $b\$(1)$, $b\$(2) \dots b\(5) の 5 個の領域を確保します。
- ② 配列の添字 0 を使わない方がわかりやすい場合があります。その場合は OPTION BASE 1 とするとメモリが無駄になりません。

【プログラム例】

```

OPTION BASE 1 _____ 添字を 1 から始める
DIM b$(5) _____ 配列 b$(1) ~ b$(5) の宣言
FOR i = 1 TO 5 _____ b$(1) ~ b$(5) にデータを読み込む
    READ b$(i)
NEXT i _____
PRINT b$(1); " "; b$(5) _____ b$(1) と b$(5) の表示
END
DATA Yamada, Osaki, Ueda, Yoshino, Honda
  
```

【結 果】

Yamada Honda

【例題 60】 2 次元配列

下のデータ組を 2 次元配列に読み表示するプログラムをつくれ。

```

データ: JAPAN      TOKYO
        USA        WASHIGTON
        UK         LONDON
        FRANCE     PARIS
        ITALY      ROME
        GERMANY    BON
  
```

【解 説】 ◎ 2 次元配列を使います。

- ① $\text{DIM } x\$(6, 2)$ は 2 次元の文字列配列 (6 行 2 列) を次のようにとります。 $6 \times 2 = 12$ 個の領域を確保します。

	0	1
0	$x\$(0, 0)$	$x\$(0, 1)$
1	$x\$(1, 0)$	$x\$(1, 1)$
2	$x\$(2, 0)$	$x\$(2, 1)$
3	$x\$(3, 0)$	$x\$(3, 1)$
4	$x\$(4, 0)$	$x\$(4, 1)$
5	$x\$(5, 0)$	$x\$(5, 1)$

```

② FOR i=0 TO 5
    FOR j=0 TO 1
        READ x$(i,j)
    NEXT j
NEXT i

```

これが2次元配列へのデータの読み込みの普通の仕方です。

③ x(6,2)$ は2次元ですが x(5,3,8,9,4)$ のように5次元の配列も使えます。最大60次元まで使えます。3次元配列の例を【例題62】、4次元の例を【例題63】で示します。

【プログラム例】

```

DIM x$(6, 2) ————— 2次元配列の宣言
FOR i = 0 TO 5 ————— 配列 x$ にデータを読み込む
    FOR j = 0 TO 1
        READ x$(i, j)
    NEXT j, i
FOR i = 0 TO 5 ————— 表示
    FOR j = 0 TO 1
        PRINT USING "&      &"; x$(i, j);
    NEXT j
PRINT
NEXT i
END
DATA JAPAN, TOKYO, USA, WASHINGTON
DATA UK, LONDON, FRANCE, PARIS
DATA ITALY, ROME, GERMANY, BON

```

【結 果】

```

JAPAN    TOKYO
USA       WASHINGTO
UK        LONDON
FRANCE    PARIS
ITALY     ROME
GERMANY   BON

```

【例題61】 数値と文字の配列

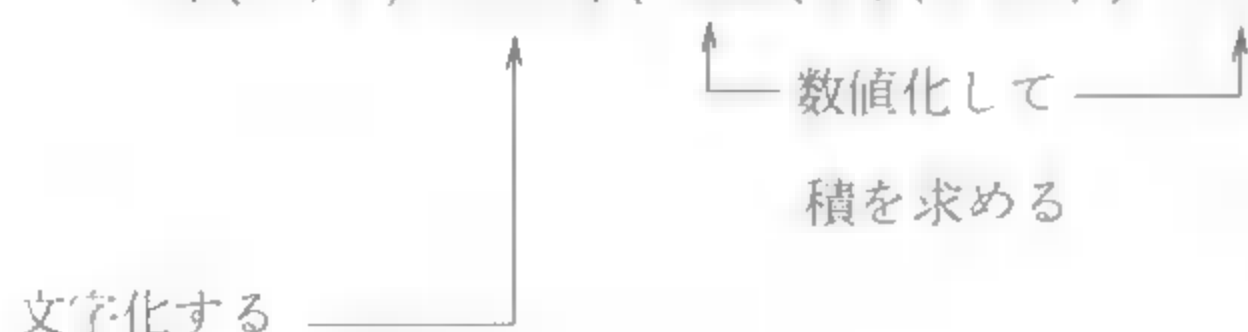
下の表を配列に代入し、金額を求めて、金額を含む表を表示するプログラムをつくれ。

商 品	数 量	単価 (千円)
パソコン	25	458
ワープロ	48	216
C D	21	142
V T R	30	98
ビデオカメラ	13	126

【解 説】 ◎ 数値と文字列の配列を考えます。

- ① 数値と文字列の混合した配列をつくることはできません。一度すべて文字として文字配列に代入し、数値としたい数字の部分は文字を数値化して使います。
 - ② 全体を文字列配列 $s\$(5,3)$ に代入します。0 列目は商品名、1 列目は数量、2 列目は単価です。ただし、3 列目に金額を代入するため宣言としては $s\$(5,4)$ とします。
 - ③ $s\$(0,0) \sim s\$(4,2)$ に文字としてデータをすべて読み込みます。
 - ④ 金額の項は第 1 列と第 2 列の積です。したがって、第 1 列と第 2 列を読み出して、それを数値化して積を求め、次にそれを文字化して 3 列目に格納します。
- たとえば、 $s\$(2,3)$ は次のように求めます。

$s\$(2,3) = \text{STR}\$(\text{VAL}(s\$(2,1)) * \text{VAL}(s\$(2,2)))$



【プログラム例】

```

DIM s$(5, 4) ————— 配列の宣言
FOR i = 0 TO 4 ————— 0～4 行についてデータ読み込みと代入
  FOR j = 0 TO 2 ————— 0～2 列を配列へ読み込む
    READ s$(i, j)
  NEXT j
  s$(i, 3) = STR$(VAL(s$(i, 1)) * VAL(s$(i, 2))) ————— 3 列の計算をして文字化して代入
NEXT i
PRINT "ショウヒン      スウリョウ      タンカ      キンカク" ————— 結果の表示
FOR i = 0 TO 4
  PRINT USING "&      & "; s$(i, 0);
  FOR j = 1 TO 3
    PRINT USING "##,###      "; VAL(s$(i, j));
  NEXT j
PRINT
NEXT i
END
DATA パソコン, 25, 458, ワーク, 48, 216, CD, 21, 142
DATA VTR, 30, 98, ビデオカメラ, 13, 126

```

【結 果】

ショウヒン	スウリョウ	タンカ	キンカク
パソコン	25	458	11,450
ワーク	48	216	10,368
CD	21	142	2,982
VTR	30	98	2,940
ビデオカメラ	13	126	1,638

【例題 62】 3次元配列

次の表を 3 次元配列に代入し、次に、地区名を入力して、その地区のデータを表示するプログラムをつくれ。

地 区	商 品	数 量	金 額	利 益	1 人当たり 売 上
東 京	パソコン	350	500	170	100
	ワープロ	650	800	120	180
大 阪	パソコン	250	300	110	120
	ワープロ	400	500	100	110
名古屋	パソコン	210	230	104	115
	ワープロ	350	450	105	109

【解 説】 ◎ 3次元配列をつくります。

- ① 3次元配列は“地区”の項，“商品”の項，“数量，金額，利益，1人当たり売上”の項の3項目です。第1の要素は“東京”，“大阪”，“名古屋”からなります。第2の要素は“パソコン”，“ワープロ”，からなります。第3の要素は“数量，金額，利益，1人当たり売上”からなります。
- ② 配列の各要素は次のようにとられます。

地 区	商 品	数 量	金 額	利 益	1 人当たり 売 上
東 京	パソコン	r% (0, 0, 0)	r% (0, 0, 1)	r% (0, 0, 2)	r% (0, 0, 3)
	ワープロ	r% (0, 1, 0)	r% (0, 1, 1)	r% (0, 1, 2)	r% (0, 1, 3)
大 阪	パソコン	r% (1, 0, 0)	r% (1, 0, 1)	r% (1, 0, 2)	r% (1, 0, 3)
	ワープロ	r% (1, 1, 0)	r% (1, 1, 1)	r% (1, 1, 2)	r% (1, 1, 3)
名古屋	パソコン	r% (2, 0, 0)	r% (2, 0, 1)	r% (2, 0, 2)	r% (2, 0, 3)
	ワープロ	r% (2, 1, 0)	r% (2, 1, 1)	r% (2, 1, 2)	r% (2, 1, 3)

配列名を r% とする

r% (x, y, z)

↑ ↑ ↑
数量が 0, 金額が 1, 利益が 2,
1 人当たり売上が 3
↑
パソコンが 0, ワープロが 1
↑
東京が 0, 大阪が 1, 名古屋が 2

- ③ 地区別が入力されたら、東京の場合は c% を 0 として、大阪の場合は c% を 1 として、名古屋の場合は c% を 2 として r% (c%, 0, 0) から r% (c%, 1, 3) までを表示します。

【プログラム例】

```

DIM r%(3, 2, 4) ————— 3次元配列の宣言
FOR i = 0 TO 2 ————— データを配列へ読み込む
  FOR j = 0 TO 1
    FOR k = 0 TO 3
      READ r%(i, j, k)
    NEXT k
  NEXT j
NEXT i

```

```

INPUT "チク ", t$ ----- 地区名の入力
IF t$ = "トウキョウ" THEN c% = 0 ----- t$の内容によってc%の値を0~2とする
IF t$ = "オオサカ" THEN c% = 1
IF t$ = "ナゴヤ" THEN c% = 2
PRINT t$ ----- 結果の表示
PRINT "      スウリョウ キンガク リエキ ウリアケ/ニン"
PRINT "ハソコン ";
PRINT r%(c%, 0, 0); " "; r%(c%, 0, 1); " "; r%(c%, 0, 2); r%(c%, 0, 3)
PRINT "ワープロ ";
PRINT r%(c%, 1, 0); " "; r%(c%, 1, 1); " "; r%(c%, 1, 2); r%(c%, 1, 3)
END
DATA 350, 500, 170, 100, 650, 800, 120, 180
DATA 250, 300, 110, 120, 400, 500, 100, 110
DATA 210, 230, 104, 115, 350, 450, 105, 109

```

【結 果】

チク ナゴヤ
ナゴヤ

ナゴヤを入力の場合

	スウリョウ	キンガク	リエキ	ウリアケ/ニン
ハソコン	210	230	104	115
ワープロ	350	450	105	109

チク トウキョウ
トウキョウ

トウキョウを入力の場合

	スウリョウ	キンガク	リエキ	ウリアケ/ニン
ハソコン	350	500	170	100
ワープロ	650	800	120	180

【例題 63】 4 次元配列

下のデータを 4 次元配列に代入し、次に、年、地区、商品名を入力して、その年のその地区のその商品の売上数量、金額、利益および 1 人当たり売上を表示するプログラムをつくれ。

年別地区別商品別売上推移

年	地 区	商品名	数 量	金 額 (万円)	利 益 (万円)	1 人当たり 売 上
1987	東 京	パソコン	100	5000	1000	500
		ワープロ	70	1400	210	470
	大 阪	パソコン	70	3200	420	530
		ワープロ	60	1200	170	400
	名古屋	パソコン	60	2980	310	600
		ワープロ	30	600	90	600
1988	東 京	パソコン	150	7400	1420	740
		ワープロ	90	1700	250	540
	大 阪	パソコン	100	4800	550	800
		ワープロ	70	1300	160	650
	名古屋	パソコン	80	3560	410	710
		ワープロ	40	750	100	750

【解 説】 ◎ 4次元の配列を扱います。

- ① 表を配列に格納するためには、第1要素は1987年と1988年の2、第2要素は東京、大阪、名古屋の3、第3要素はパソコンとワープロの2、第4要素は数量、金額、利益、1人当たり売上の4です。したがって、次のように宣言します。ただし、

```
DIM h%(1 TO 2, 1 TO 3, 1 TO 2, 1 TO 4)
```

これはOPTION BASE 1とすればDIM h%(2, 3, 2, 4)と同じです。

- ② h%(2, 3, 1, 2)は1988年の名古屋のパソコンの金額を示します。
 ③ データの読み込みは次のようにします。

```
FOR i=1 TO 2
  FOR j=1 TO 3
    FOR k=1 TO 2
      FOR l=1 TO 4
        READ h%(i, j, k, l)
      NEXT l, k, j, i
```

【プログラム例】

```
DIM h%(1 TO 2, 1 TO 3, 1 TO 2, 1 TO 4)——4次元配列の宣言
FOR i = 1 TO 2——データの読み込み
  FOR j = 1 TO 3
    FOR k = 1 TO 2
      FOR l = 1 TO 4
        READ h%(i, j, k, l)
      NEXT l, k, j, i
  INPUT "ネソ "; y%——年、地区、商品名の入力
  INPUT "チク "; t$
  INPUT "ショウヒソ "; s$
  IF y% = 1987 THEN i = 1 ELSE i = 2——年が1987のときiを1、その他はiを2とする
  IF t$ = "トウキョウ" THEN——地区が東京のときjを1、大阪のときjを2、
    j = 1——その他のときjを3とする
  ELSEIF t$ = "オオサカ" THEN
    j = 2
  ELSE
    j = 3
  END IF
  IF s$ = "ハソコソ" THEN k = 1 ELSE k = 2——商品名がパソコンのときkを1、その他のときkを2とする
  FOR m = 1 TO 4——i, j, kを各要素の値として数量、金額、利益、1人当たり
    PRINT h%(i, j, k, m);——売上を表示
  NEXT m
END
DATA 100, 5000, 1000, 500, 70, 1400, 210, 470
DATA 70, 3200, 420, 530, 60, 1200, 170, 400
DATA 60, 2980, 310, 600, 30, 600, 90, 600
DATA 150, 7400, 1420, 740, 90, 1700, 250, 540
DATA 100, 4800, 550, 800, 70, 1300, 160, 650
DATA 80, 3560, 410, 710, 40, 750, 100, 750
```

【結 果】

```

ネン ? 1988
チク ? オオサカ
ショウヒン ? ハ°ソコン
100 4800 550 800

```

1988, オオサカ, パソコンを入力の例

```

ネン ? 1987
チク ? トウキョウ
ショウヒン ? ワ°プロ
70 1400 210 470

```

1987, トウキョウ, ワ°プロを入力の例

〔演習問題〕

- (64) 配列 a (20) に下のデータを読み、2 行に表示するプログラムをつくれ。

```

データ;3.5, 2.5, 1.5, 3.3, 4.2, 5.2, 2.6, 3.5, 4.2,
        8.1, 1.5, 2.3, 5.6, 8.2, 9.9, 5.4, 6.3, 2.2,
        1.8, 4.4

```

- (65) x を -10 から 10 まで
- $y=3x^3-2x^2+5x$
- の値を求めて配列 y%(-10)~y%(10) に代入してから、各値を表示するプログラムをつくれ。

- (66) 下のデータを配列に読み a%(5) と a%(7) を表示するプログラムをつくれ。ただし、配列の添字は 1 からとるものとする。

```

データ;7, 5, 2, 9, 7, 3, 8, 4, 1, 0

```

- (67) オプションベースを 1 として、配列 x!(10,20) を宣言し、縦方向 10、横方向 20 の表と考え乱数をつくって代入し、x!(5,12) の位置の値を表示するプログラムをつくれ。

- (68) 下の表を 2 次元配列に読み表示するプログラムをつくれ。

5	3	4
2	3	8
7	4	5
6	1	8

- (69) 下の表の相対する項目の積の値を配列に代入して、表示するプログラムをつくれ。

8	7	6	7	4	9
2	9	3	5	3	1
4	8	2	6	8	2
7	1	0	3	2	7

- (70) 【例題 61】のデータを使い、配列に値を読み込んでから、5 品目の金額の合計を求めるプログラムをつくれ。

- (71) 乱数で 100 個の 10~99 の値をつくり、10~90 台が各々が何回おきたかを求めるプログラムをつくれ。

(72) 下の文字データを配列に読み、後から表示するプログラムをつくれ。

データ;Komoro, Karuizawa, Nagano, Ueda, Ueno

(73) (72) の文字列配列を使って、先頭 1 文字を入力して、その文字で始まる文字列を表示するプログラムをつくれ。

(74) 【例題 62】のデータを使い 3 次元配列に値を代入して、商品番号 1 または 2 を入力し、1 のときは“パソコン”のデータ、2 のときは“ワープロ”のデータを表示するプログラムをつくれ。

(75) 【例題 62】のデータを使い 3 次元配列に値を代入してから金額を入力し、その金額以上の利益の地区と品名のデータを表示するプログラムをつくれ。

〔解 答〕

(64) 【プログラム例】

```

DIM a(20) ————— 配列 a (20) の宣言
FOR i = 0 TO 19 ————— a(0)~a(19)にデータを読み込む
    READ a(i)
NEXT i
FOR i = 0 TO 1 ————— a(0)~a(19)を2行にわけて表示
    FOR j = 0 TO 9
        PRINT a(10 * i + j); ————— a(0)~a(9)が1行目, a(10)~a(19)が2行目
    NEXT j
    PRINT
NEXT i
END
DATA 3.5, 2.5, 1.5, 3.3, 4.2, 5.2, 2.6, 3.5, 4.2, 8.1
DATA 1.5, 2.3, 5.6, 8.2, 9.9, 5.4, 6.3, 2.2, 1.8, 4.4
    
```

【結 果】

```

3.5  2.5  1.5  3.3  4.2  5.2  2.6  3.5  4.2  8.1
1.5  2.3  5.6  8.2  9.9  5.4  6.3  2.2  1.8  4.4
    
```

(65) 【プログラム例】

```

DIM y%(-10 TO 10) ————— 配列 y%(-10)~y%(10)の宣言
FOR x = -10 TO 10 ————— x を-10 から 10 まで代入して
    y%(x) = 3 * x * x * x - 2 * x * x + 5 * x ————— y=3x3-2x2+5x を求めて配列へ代入
NEXT x
FOR x = -10 TO 10 ————— y%(-10)~y%(10)の表示
    PRINT y%(x)
NEXT x
END
    
```

【結 果】

```

-3250
-2394
-1704
-1162
-750
    
```

```

-450
-244
-114
-42
-10
0
6
26
78
180
350
606
966
1448
2070
2850

```

(66) 【プログラム例】

```

OPTION BASE 1 ————— 添字を 1 から始める
DIM a%(10) ————— 配列 a%(1)~a%(10)の宣言
FOR i = 1 TO 10 ————— a%(1)~a%(10)にデータを読み込む
    READ a%(i)
NEXT i
PRINT a%(5); a%(7) ————— a%(5) と a%(7)の表示
END
DATA 7, 5, 2, 9, 7, 3, 8, 4, 1, 0

```

【結 果】

7 8

(67) 【プログラム例】

```

OPTION BASE 1 ————— オプションベースを 1 とする
DIM x!(10, 20) ————— 配列 x!(10,20)の宣言
FOR i = 1 TO 10 ————— x!(1,1)から x!(10,20)に乱数を代入
    FOR j = 1 TO 20
        x!(i, j) = RND(1)
    NEXT j
NEXT i
PRINT x!(5, 12) ————— x!(5,12)の値を表示
END

```

【結 果】

.46298

(68) 【プログラム例】

```

DIM x%(4, 3) ————— 2次元配列の宣言
FOR i = 0 TO 3 ————— 配列 x%にデータを読み込む
    FOR j = 0 TO 2
        READ x%(i, j)
    NEXT j
NEXT i

```



```

FOR i = 0 TO 3
  FOR j = 0 TO 2
    PRINT x%(i, j);
  NEXT j
PRINT
NEXT i
END
DATA 5, 3, 4, 2, 3, 8, 7, 4, 5, 6, 1, 8

```

表示

【結 果】

5	3	4
2	3	8
7	4	5
6	1	8

(69) 【プログラム例】

```

DIM s%(4, 3), t%(4, 3), u%(4, 3)
FOR i = 0 TO 3
  FOR j = 0 TO 2
    READ s%(i, j), t%(i, j)
    u%(i, j) = s%(i, j) * t%(i, j)
  NEXT j
NEXT i
FOR i = 0 TO 3
  FOR j = 0 TO 2
    PRINT USING "### "; s%(i, j);
  NEXT j
  PRINT " ";
  FOR j = 0 TO 2
    PRINT USING "### "; t%(i, j);
  NEXT j
  PRINT " ";
  FOR j = 0 TO 2
    PRINT USING "### "; u%(i, j);
  NEXT j
PRINT
NEXT i
END
DATA 8, 7, 7, 4, 6, 9, 2, 5, 9, 3, 3, 1, 4, 6, 8, 8, 2, 2, 7, 3, 1, 2, 0, 7

```

配列の宣言

2つの表を配列 s%, t%へ読み込み相対する値の積を求めて配列 u%の相対する項へ代入

結果の表示

【結 果】

8	7	6	7	4	9	56	28	54
2	9	3	5	3	1	10	27	3
4	8	2	6	8	2	24	64	4
7	1	0	3	2	7	21	2	0

(70) 【プログラム例】

```

DIM s$(5, 4)
FOR i = 0 TO 4
  FOR j = 0 TO 2

```

配列の宣言

データの読み込みと3列目の計算をして代入および合計を求める

```

    READ s$(i, j)
  NEXT j
  s$(i, 3) = STR$(VAL(s$(i, 1)) * VAL(s$(i, 2)))
  t = t + VAL(s$(i, 3))
NEXT i
PRINT t
END
DATA ハッソソ, 25, 458, ワープロ, 48, 216, CD, 21, 142
DATA VTR, 30, 98, ビデオカメラ, 13, 126

```

金額の合計を求める

結果の表示

【結 果】

29378

(71) 【プログラム例】

```

DIM s%(1 TO 9)
FOR i = 1 TO 100
  p = INT(RND(3) * 90) + 10
  pp = INT(p / 10)
  s%(pp) = s%(pp) + 1
NEXT i
FOR i = 1 TO 9
  PRINT i * 10; "タ^イ="; s%(i)
NEXT i
END

```

配列の宣言

乱数をつくり数える

10~99の乱数

乱数の10の値をとり出す

乱数の値によってs%(1)~s%(9)に代入

結果の表示

【結 果】

```

10 タ^イ= 14
20 タ^イ= 4
30 タ^イ= 16
40 タ^イ= 13
50 タ^イ= 9
60 タ^イ= 12
70 タ^イ= 10
80 タ^イ= 9
90 タ^イ= 13

```

(72) 【プログラム例】

```

DIM a$(1 TO 5)
FOR i = 1 TO 5
  READ a$(i)
NEXT i
FOR i = 5 TO 1 STEP -1
  PRINT a$(i); " ";
NEXT i
PRINT
END
DATA Komoro, Karuizawa, Nagano, Ueda, Ueno

```

配列の宣言

データの読み込み

逆に表示

【結 果】

Ueno Ueda Nagano Karuizawa Komoro

(73) 【プログラム例】

```

DIM a$(1 TO 5) ————— 配列の宣言
FOR i = 1 TO 5 ————— データの読み込み
    READ a$(i)
NEXT i
INPUT x$ ————— 文字列の入力
FOR i = 1 TO 5 ————— x$で始まる文字列の表示
    IF ASC(a$(i)) = ASC(x$) THEN PRINT a$(i); " ";
NEXT i
PRINT
END
DATA Komoro, Karuizawa, Nagano, Ueda, Ueno

```

【結 果】

```

? U
Ueda Ueno
? K
Komoro Karuizawa

```

(74) 【プログラム例】

```

DIM r%(3, 2, 4) ————— 配列の宣言
FOR i = 0 TO 2 ————— データの読み込み
    FOR j = 0 TO 1
        FOR k = 0 TO 3
            READ r%(i, j, k)
        NEXT k
    NEXT j
NEXT i
INPUT "ショウヒン NO (1 or 2) ", t% ————— 商品番号の入力
IF t% = 1 THEN s$ = "ハソコソ" ————— 商品番号によって s$を決める
IF t% = 2 THEN s$ = "ワフコソ"
c% = t% - 1 ————— 商品番号によって商品を示す c%を決める
t1$(0) = "トウキョウ" ————— 地区名を配列へ代入
t1$(1) = "オオサカ"
t1$(2) = "ナゴヤ"
PRINT s$ ————— 結果の表示
PRINT "      スウリョウ  キンガク  リエキ  ウリアゲ / ニン"
FOR i = 0 TO 2
    PRINT t1$(i);
    PRINT r%(i, c%, 0); " "; r%(i, c%, 1); " "; r%(i, c%, 2); r%(i, c%, 3)
NEXT i
END
DATA 350, 500, 170, 100, 650, 800, 120, 180
DATA 250, 300, 110, 120, 400, 500, 100, 110
DATA 210, 230, 104, 115, 350, 450, 105, 109

```

【結 果】

```

ショウヒン NO (1 or 2) 1
ハソコソ
      スウリョウ  キンガク  リエキ  ウリアゲ / ニン
トウキョウ  350      500      170      100

```

1 を入力の場合

オオサカ	250	300	110	120
ナコ`ヤ	210	230	104	115
シヨウヒン	NO (1 or 2) 2			
ワ-フ`ロ				
	スウリヨウ	キンカ`ク	リエキ	ウリアケ`/ニン
トウキヨウ	650	800	120	180
オオサカ	400	500	100	110
ナコ`ヤ	350	450	105	109

2 を入力の場合

(75) 【プログラム例】

```

DIM r%(3, 2, 4) ————— 配列の宣言
FOR i = 0 TO 2 ————— データの読み込み
  FOR j = 0 TO 1
    FOR k = 0 TO 3
      READ r%(i, j, k)
    NEXT k
  NEXT j
NEXT i
INPUT "キンガ`ク ", k% ————— 金額の入力
t1$(0) = "トウキヨウ" ————— 地区名を配列へ代入
t1$(1) = "オオサカ"
t1$(2) = "ナコ`ヤ"
h$(0) = "ハ`ソコン" ————— 商品名を配列へ代入
h$(1) = "ワ-フ`ロ"
PRINT k%; "イジ`ョウ" ————— 結果の表示
PRINT "チク   ヒンメイ   リエキ"
FOR i = 0 TO 2
  IF r%(i, 0, 2) >= k% THEN PRINT t1$(i); h$(0); r%(i, 0, 2)
  IF r%(i, 1, 2) >= k% THEN PRINT t1$(i); h$(1); r%(i, 1, 2)
NEXT i
END
DATA 350, 500, 170, 100, 650, 800, 120, 180
DATA 250, 300, 110, 120, 400, 500, 100, 110
DATA 210, 230, 104, 115, 350, 450, 105, 109

```

【結 果】

キンカ`ク 110	110 を入力の場合
110 イジ`ョウ	
チク ヒンメイ リエキ	
トウキヨウ ハ`ソコン 170	
トウキヨウ ワ-フ`ロ 120	
オオサカ ハ`ソコン 110	
キンカ`ク 105	105 を入力の場合
105 イジ`ョウ	
チク ヒンメイ リエキ	
トウキヨウ ハ`ソコン 170	
トウキヨウ ワ-フ`ロ 120	
オオサカ ハ`ソコン 110	
ナコ`ヤ ワ-フ`ロ 105	
キンカ`ク 130	130 を入力の場合
130 イジ`ョウ	
チク ヒンメイ リエキ	
トウキヨウ ハ`ソコン 170	

9 章 タイプ型

TYPE, END TYPE

【例題 64】 ユーザー定義型

下のデータをユーザー定義型（タグを KOZIN とし、名前を 10 文字、年齢を整数型、年収を単精度実数型、出身地を 7 文字とするメンバー）で定義し、各項名を表示するプログラムをつくれ。

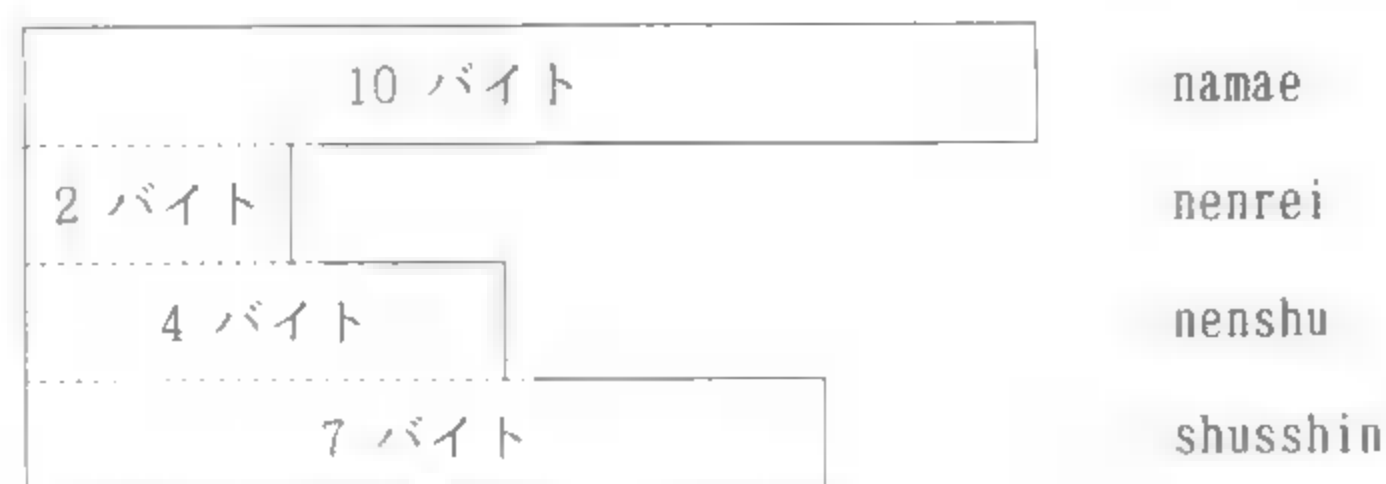
データ ; Onoda, 50, 1350.6, トウキョウト
(名前, 年齢, 年収 (万円), 出身地の順とする)

【解 説】 ① ユーザー定義型（データ型）を使います。

- ① ユーザー定義型は C 言語の構造型に相当するもので、従来の BASIC にはない新しいデータ型です。本例では名前、年齢、年収、出身地の組合わせのデータを 1 つのかたまりとして扱おうとするものです。
- ② ユーザー定義型は次のように定義します。

TYPE kozin	……全体を kozin という名前とします。タグと呼ばれます。	
namae AS STRING*10	……namae をメンバー名とし、固定長 10 文字とします	メンバー
nenrei AS INTEGER	……nenrei をメンバー名とし、整数型とします。	
nenshu AS SINGLE	……nenshu をメンバー名とし、単精度実数とします	
shusshin AS STRING*7	……shusshin をメンバー名とし、固定長 7 文字とします	
END TYPE	……ユーザー定義型の定義の終了	

この結果次のような 1 かたまりの領域が確保されます。文字型、整数型、実数型、文字型と型の違うものを 1 かたまりとして扱います。



- ③ このタグ `kozin` の変数は次のように宣言します。

レコード型変数名
`DIM a AS kozin`

- ④ 各メンバーを参照（各メンバーに値を代入したり、読み出したりすること）するには次のように行います。

```
a.namae = "Onoda"
a.nenrei = 50
a.nenshu = 1350.6
a.shusshin = "トウキョウ"
```

- ⑤ 表示は `PRINT a.namae` で "Onoda" を表示します。

- ⑥ メンバーは配列と可変型の文字列（たとえば、`x$` という型。これは `x$` の内容が "ABC" と 3 文字であったり "ABCDE" と 5 文字であったりします）にはなりません。

【プログラム例】

```
TYPE kozin
    namae AS STRING * 10
    nenrei AS INTEGER
    nenshu AS SINGLE
    shusshin AS STRING * 7
END TYPE

DIM a AS kozin
a.namae = "Onoda"
a.nenrei = 50
a.nenshu = 1350.6
a.shusshin = "トウキョウ"

PRINT a.namae
PRINT a.nenrei
PRINT a.nenshu
PRINT a.shusshin
END
```

ユーザー定義型の宣言
 メンバー `namae` は 10 文字の文字列
 メンバー `nenrei` は整数
 メンバー `nenshu` は実数
 メンバー `shusshin` は 7 文字の文字列
 変数名 `a` の宣言
 変数にデータを入力
 各変数の表示

【結 果】

```
Onoda
50
1350.6
トウキョウ
```


【例題 65】 ユーザー定義型での配列

下表のデータをタグを `shain` としてユーザー定義型で定義して、配列にデータを読み、全データを表示するプログラムをつくれ。

名 前	年 収	家族数
Yamada	1500	6
Tanaka	1800	4
Onoda	850	3
Kawada	1200	4
Noda	1000	5
Hoshino	750	3

【解 説】 ◎ 変数に配列を使います。

- ① データ組が複数の場合は変数に配列を使います。
- ② 【例題 64】と同様にタグを `shain` としてメンバーを `namae`, `nenshu`, `kazokusu` として定義します。
- ③ 変数の定義を次のようにします。この結果、0～5 の 6 組のデータを扱えます。
`DIM a(6) AS shain`
- ④ 参照は `a(1).namae`, `a(3).nenshu`, `a(4).kazokusu` のように配列名・メンバー名で行います。

【プログラム例】

```

TYPE shain ————— ユーザー定義型の定義
    namae AS STRING * 10
    nenshu AS INTEGER
    kazokusu AS INTEGER
END TYPE
DIM a(6) AS shain ————— 変数の宣言
FOR i = 0 TO 5 ————— データの読み込み
    READ a(i).namae
    READ a(i).nenshu
    READ a(i).kazokusu
NEXT i
FOR i = 0 TO 5 ————— 表示
    PRINT USING "&      &"; a(i).namae;
    PRINT USING "#####   "; a(i).nenshu; a(i).kazokusu
NEXT i
END
DATA Yamada, 1500, 6, Tanaka, 1800, 4, Onoda, 850, 3
DATA Kawada, 1200, 4, Noda, 1000, 5, Hoshino, 750, 3

```

【結 果】

Yamada	1500	6
Tanaka	1800	4
Onoda	850	3
Kawada	1200	4
Noda	1000	5
Hoshino	750	3

【例題 66】 タイプ型を使って検索

次のデータをタイプ型で定義して代入し、次に、身長を入力してその身長以上の人のリストをつくるプログラムをつくれ。

名 前	身長 (cm)	体重 (kg)	年 齢
神田国男	183.7	65.8	41
大友良雄	160.6	54.2	28
中根富夫	158.9	50.6	36
井上太郎	190.3	72.8	29
吉田 薫	181.3	83.9	33
渡辺兼介	175.2	59.2	42
新井久男	170.6	70.3	38

【解 説】 ① タイプ型変数を使って検索をします。

① 問題のデータを次の名前でタイプ型変数に代入します。

```
TYPE people
```

namae AS STRING*20	名前 20 バイト
shinchou AS SINGLE	身長 単精度実数
taiju AS SINGLE	体重 単精度実数
nenrei AS INTEGER	年齢 整数

```
END TYPE
```

② データを配列 a(i) に代入します。

③ 身長を shi に代入して、変数名 a(i).shinchou と比較して、shi 以上の人のリストを表示します。

【プログラム例】

```

TYPE people
  namae AS STRING * 20
  shinchou AS SINGLE
  taiju AS SINGLE
  nenrei AS INTEGER
END TYPE
DIM a(7) AS people

```

タイプ型変数の宣言
名前、身長、体重、年齢を代入する準備

配列変数の宣言

```

FOR i = 0 TO 6
    READ a(i).namae
    READ a(i).shinchou
    READ a(i).taiju
    READ a(i).nenrei
NEXT i
INPUT "シンチョウ ", shi
FOR i = 0 TO 6
    IF a(i).shinchou >= shi THEN PRINT a(i).namae
NEXT i
END
DATA カンダ クニオ,183.7,65.8,41
DATA オオトモ ヨシオ,160.6,54.2,28
DATA ナカネ トミオ,158.9,50.6,36
DATA イノウエ タロウ,190.3,72.8,29
DATA ヨシダ カオル,181.3,83.9,33
DATA ワタナベ ケンスケ,175.2,59.2,42
DATA アライ ヒサオ,170.6,70.3,38
    
```

配列ヘデータ読み込み

身長入力

shi 以上の人の名前を表示
180.6 を入力の例

【結 果】

```

シンチョウ 180.6
カンダ クニオ
イノウエ タロウ
ヨシダ カオル
    
```

【例題 67】 全体のコピー

次のデータ組を、タグを shain とするユーザー定義型で定義し、変数 a と b を宣言し、a.namae, a.nenshu, a.kazokusu に代入した後、b へコピーして、b の各メンバーの値を表示するプログラムをつくれ。

データ :

yamada	1500	6
--------	------	---

 名前、年収、家族数とする

【解 説】 ① ユーザー定義型の全データを変数間でコピーします。

- ① タグを shain とし、メンバーを namae, nenshu, kazokusu として定義します。
- ② 次に変数 a と b を宣言します。
- ③ 各変数にデータを読み込みます。
- ④ b=a によって a のすべてのメンバーを b に代入します。ユーザー定義型のすべてのデータを変数間でコピーできる便利さがあります。

【プログラム例】

```

TYPE shain
    namae AS STRING * 10
    nenshu AS INTEGER
    kazokusu AS INTEGER
END TYPE
DIM a AS shain
DIM b AS shain
    
```

ユーザー定義型の宣言

変数 a, b の宣言

```

READ a.namae
READ a.nenshu
READ a.kazokusu
b = a
PRINT USING "&      &"; b.namae;
PRINT USING "#####   ###"; b.nenshu; b.kazokusu
END
DATA Yamada, 1500, 6

```

変数 a へデータを読み込む

a を b へコピー

b の表示

【結 果】

Yamada 1500 6

〔演習問題〕

- (76) タグを `kaisha` とするユーザー定義型で `shamei`, `uriage`, `jugyoin` をメンバーとして定義して、次に変数名を `b` として、下のデータを代入してから表示するプログラムをつくれ。

データ; `Nihon Seisakusho, 1536.2, 1200` (社名, 売上 (億円), 従業員 (人)) の順とする。

- (77) 下のデータ組を、ユーザー定義型で定義し、変数 `a` と `b` を宣言し、`a` にデータ組を代入し、次に全体を `b` へコピーしてから `b` を表示するプログラムをつくれ。

データ:

大山	東京都	3
----	-----	---

 名前, 住所, 家族数とする

- (78) 次のデータを、ユーザー定義型で定義し、`a(4)` を宣言して代入した後、全体を表示するプログラムをつくれ。

大山	東京都	3
上田	埼玉	4
飯田	千葉	5
豊島	神奈川	4

- (79) 下のデータをユーザー定義型で定義して、次に名前を入力して、その名前の人のデータを表示するプログラムをつくれ。

名 前	年齢	所 属	特 技	勤続年数	身長 (cm)	体重 (kg)
大山和男	31	営業 1 課	英会話	7	163.8	60.3
木下三郎	28	総務課	パソコン	3	156.9	51.4
上田次郎	19	営業 3 課	音 楽	1	172.8	62.5
玉井久雄	24	経理課	英会話	2	178.3	70.6
川上太郎	33	営業 2 課	パソコン	5	181.5	75.3
大下一夫	25	企画課	書 道	4	170.2	63.6

- (80) (79) のデータを使って勤続年数を入力し、その年数以上の人のリストを表示するプログラムをつくれ。

〔解 答〕

(76) 【プログラム例】

```

TYPE kaisha
  shamei AS STRING * 20
  uriage AS SINGLE
  jugyoin AS INTEGER
END TYPE
DIM b AS kaisha
b.shamei = "Nihon Seisakusho"
b.uriage = 1536.2
b.jugyoin = 1200
PRINT b.shamei
PRINT b.uriage; "オクエン"
PRINT b.jugyoin; "ニン"
END

```

ユーザー定義型の宣言
変数名を b とする
代入
表示

【結 果】

```

Nihon Seisakusho
1536.2 オクエン
1200 ニン

```

(77) 【プログラム例】

```

TYPE meibo
  namae AS STRING * 10
  jusho AS STRING * 8
  kazokusu AS INTEGER
END TYPE
DIM a AS meibo
DIM b AS meibo
READ a.namae
READ a.jusho
READ a.kazokusu
b = a
PRINT b.namae; b.jusho; b.kazokusu
END
DATA オヤマ, トウキョウト, 3

```

ユーザー定義型の宣言
変数 a, b の宣言
変数に値を読み込む
b に a をコピー
表示

【結 果】

```

オオヤマ      トウキョウト      3

```

(78) 【プログラム例】

```

TYPE meibo
    namae AS STRING * 10
    jusho AS STRING * 8
    kazokusu AS INTEGER
END TYPE
DIM a(4) AS meibo
FOR i = 0 TO 3
    READ a(i).namae
    READ a(i).jusho
    READ a(i).kazokusu
NEXT i
FOR i = 0 TO 3
    PRINT a(i).namae; a(i).jusho; a(i).kazokusu
NEXT i
END
DATA オオヤマ, トウキョウト, 3, ウエダ, サイトマ, 4
DATA イタ, チハ, 5, トシマ, カナガワ, 4

```

ユーザー定義型の宣言

変数名を a(4) と配列とする

変数に値を読み込む

表示

【結 果】

オオヤマ	トウキョウト	3
ウエダ	サイトマ	4
イタ	チハ	5
トシマ	カナガワ	4

(79) 【プログラム例】

```

TYPE shain
    namae AS STRING * 8
    nenrei AS INTEGER
    shozoku AS STRING * 10
    tokugi AS STRING * 10
    kinzoku AS INTEGER
    shinchou AS SINGLE
    taiju AS SINGLE
END TYPE
DIM a(6) AS shain
FOR i = 0 TO 5
    READ a(i).namae
    READ a(i).nenrei
    READ a(i).shozoku
    READ a(i).tokugi
    READ a(i).kinzoku
    READ a(i).shinchou
    READ a(i).taiju
NEXT i
INPUT "名前 ", na$
FOR i = 0 TO 5
    IF a(i).namae = na$ THEN
        PRINT a(i).namae; a(i).nenrei; a(i).shozoku; a(i).tokugi;
        PRINT a(i).kinzoku; a(i).shinchou; a(i).taiju
    END IF

```

タイプ型変数宣言
名前, 年齢, 所属, 特技, 勤続, 身長, 体重

配列の宣言

データを配列へ読み込む

名前の入力

na\$と同じ名前の人のデータの表示


```

NEXT i
END
DATA 大山和男, 31, 営業一課, 英会話, 7, 163.8, 60.3
DATA 木下三郎, 28, 総務課, パソコン, 3, 156.9, 51.4
DATA 上田次郎, 19, 営業三課, 音楽, 1, 172.8, 62.5
DATA 玉井久雄, 24, 経理課, 英会話, 2, 178.3, 70.6
DATA 川上太郎, 33, 営業二課, パソコン, 5, 181.5, 75.3
DATA 木下一夫, 25, 企画課, 書道, 4, 170.2, 63.6

```

【結 果】

```

名前 木下三郎
木下三郎 28 総務課      パソコン      3   156.9   51.4      木下三郎を入力の例

```

(80) 【プログラム例】

```

TYPE shain
  namae AS STRING * 8
  nenrei AS INTEGER
  shozoku AS STRING * 10
  tokugi AS STRING * 10
  kinzoku AS INTEGER
  shinchou AS SINGLE
  taiju AS SINGLE
END TYPE
DIM a(6) AS shain
FOR i = 0 TO 5
  READ a(i).namae
  READ a(i).nenrei
  READ a(i).shozoku
  READ a(i).tokugi
  READ a(i).kinzoku
  READ a(i).shinchou
  READ a(i).taiju
NEXT i
INPUT "勤続年数 ", ne
FOR i = 0 TO 5
  IF a(i).kinzoku >= ne THEN PRINT a(i).namae
NEXT i
END
DATA 大山和男, 31, 営業一課, 英会話, 7, 163.8, 60.3
DATA 木下三郎, 28, 総務課, パソコン, 3, 156.9, 51.4
DATA 上田次郎, 19, 営業三課, 音楽, 1, 172.8, 62.5
DATA 玉井久雄, 24, 経理課, 英会話, 2, 178.3, 70.6
DATA 川上太郎, 33, 営業二課, パソコン, 5, 181.5, 75.3
DATA 木下一夫, 25, 企画課, 書道, 4, 170.2, 63.6

```

タイプ型
名前、年齢、所属、特技、勤続、身長、体重

配列宣言

データの読み込み

勤続年数の入力

勤続年数が **ne** 以上の人をリスト

【結 果】

```

勤続年数 4      4 を入力の場合
大山和男
川上太郎
木下一夫

```

10 章 式の定義

DEF FN

【例題 68】 DEF

円の面積を求める式 $3.14159 * r * r$ (ただし r は半径) を定義して、次に半径 r を入力して、半径 r の円の面積を求めるプログラムをつくれ。

【解 説】 ◎ DEF により式を定義します。

- ① DEF $fna(r) = 3.14159 * r * r$ によって $3.14159 * r * r$ を $fna(r)$ という名前で定義します。
fn の次の a が変数名で最大 40 文字まで使えます。
- ② () 内の r がパラメータで r の値が与えられると $3.14159 * r * r$ の値を $fna(r)$ が持ちます。たとえば r を 5 とすると $fna(5)$ は $3.14159 \times 5 \times 5$ となります。
- ③ DEF FN 関数は定義されたモジュールの中でのみ使えます。他のモジュールからの参照はできません。
- ④ DEF $fna(r)$ の型は単精度実数型です。DEF $fna!(r)$ と明示することができます。

【プログラム例】

```
DEF fna (r) = 3.14159 * r * r —— 式の定義
INPUT "r "; r —— 半径 r の入力
s = fna(r) —— 式の値を求め s に代入
PRINT "ハンケイ"; r; " / メンセキ="; s —— 結果の表示
END
```

【結 果】

```
r ? 123 —— 123 を入力の例
ハンケイ 123 ノ メンセキ= 47529.12
r ? 5.8 —— 5.8 を入力の例
ハンケイ 5.8 ノ メンセキ= 105.6831
```

【例題 69】 複数のパラメータ

$ax^2 + bx + c$ を関数で定義し、次に a , b , c , x の値を代入して、 $ax^2 + bx + c$ の値を求めるプログラムをつくれ。

【解 説】 ◎ 複数のパラメータを使います。

- ① DEF $fna(a, b, c, x) = a * x * x + b * x + c$ では $ax^2 + bx + c$ を定義します。このときのパラメータは a , b , c , x の 4 つです。

【プログラム例】

```

DEF fna (a, b, c, x) = a * x * x + b * x + c ----- a2+bx+c の定義
INPUT "a,b,c,x "; a, b, c, x ----- a, b, c, x の入力
y = fna(a, b, c, x) ----- 式の値を求め y に代入
IF b >= 0 THEN z1$ = "+" ELSE z1$ = "-" ----- +, -のときの記号の表示のための処理
IF c >= 0 THEN z2$ = "+" ELSE z2$ = "-" -----
PRINT a; "*"; x; "^2"; z1$; b; "*"; x; z2$; c; "="; y ----- 結果の表示
END

```

【結 果】

```

a,b,c,x ? 3,-5,2,3 ----- 3, -5, 2, 3, の入力の例
3 * 3 ^2 -5 * 3 + 2 = 14
a,b,c,x ? 2,5,4,2 ----- 2, 5, 4, 2 を入力の例
2 * 2 ^2 + 5 * 2 + 4 = 22
a,b,c,x ? -3,7,-6,2 ----- -3, 7, -6, 2 を入力の例
-3 * 2 ^2 + 7 * 2 -6 = -4
a,b,c,x ? 2,5,3,-4 ----- 2, 5, 3, -4 を入力の例
2 * -4 ^2 + 5 * -4 + 3 = 15

```

【例題 70】 文字列を値とする関数式

"Microsoft[" + a\$ を文字列とする式を定義し、次に文字列を入力して、Microsoft とつないで表示するプログラムをつくれ。

【解 説】 ◎ 文字列を値とする関数式を定義します。

- ① DEF fnx\$(a\$)="Microsoft[" + a\$ の定義によって "Microsoft[" と文字列 a\$ をつないだ文字列を式 fnx\$(a\$) の値とします。

【プログラム例】

```

DEF fnx$ (a$) = "Microsoft " + a$ ----- 式の定義
INPUT x$ ----- x$ の入力
y$ = fnx$(x$) ----- 式の値を y$ に代入
PRINT y$ ----- 結果の表示
END

```

【結 果】

```

? Quick C ----- Quick C を入力の例
Microsoft Quick C
? Quick Basic ----- Quick Basic を入力の例
Microsoft Quick Basic

```

【例題 71】 関数式の演算

a^3 と b^2 を求める式を各々 fnx(a) と fnb(b) で定義し、次に、x と y を入力し fnx(x)/fnb(y) の値を求めるプログラムをつくれ。

【解 説】 ◎ 関数式の演算を行います。

- ① 下の定義で関数 fnx と fnb を定義します。

```
DEF fnx(a)=a*a*a
```

```
DEF fnb(b)=b*b
```

- ② x と y を入力し, $z=fnx(x)/fnb(y)$ を行うと x^3/y^2 を求めて z に代入します.
- ③ もちろん $DEF fnx(a,b)=a*a*a/(b*b)$ によって一度に式を定義できますが a^3 と b^2 を別のところで使いたい場合は別々に定義する必要があります.

【プログラム例】

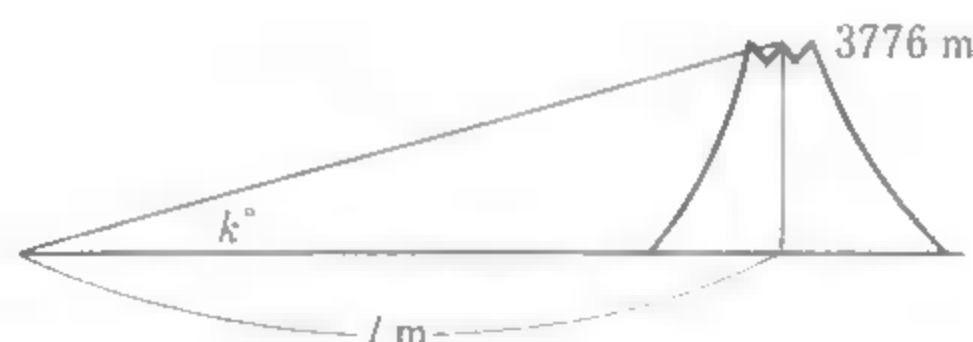
```
DEF fnx (a) = a * a * a ——  $a^3$  の式の定義
DEF fnb (b) = b * b ——  $b^2$  の式の定義
INPUT x, y ——  $x, y$  の入力
z = fnx(x) / fnb(y) ——  $x^3/y^2$  を求めて  $z$  に代入
PRINT x; "^3 / "; y; "^2="; z —— 結果の表示
END
```

【結 果】

```
? 6.3,2.4 —— 6.3, 2.4 を入力の例
6.3 ^3 / 2.4 ^2= 43.41094
? 5,3 —— 5, 3 を入力の例
5 ^3 / 3 ^2= 13.88889
? 4,2 —— 4, 2 を入力の例
4 ^3 / 2 ^2= 16
```

【例題 72】 算術関数を使った関数式

下図の角度 k を入力して, 距離 l を求めるプログラムをつくれ.



【解 説】 ◎ 関数定義の中で算術関数を使います.

- ① 関数定義の中で, Quick BASIC の持つ関数を使うことができます. 次の定義は l を求める式です. 算術関数 TAN (正接) を使っています.

```
DEF fna(k)=3776/TAN(k/180*3.14159)
```

↑ ↑ ↑
 k 度のときの l を 正接 k 度をラジアンに変換
求める式の定義

【プログラム例】

```
DEF fna (k) = 3776 / TAN(k / 180 * 3.14159) —— 式の定義
INPUT "k "; k ——  $k$  度の入力
l = fna(k) ——  $l$  を求める
PRINT "k"; k; " のとき / キロ="; l; " m" —— 結果の表示
END
```

【結 果】

```

k ? 10 ----- 10 を入力の場合
カク 10 トノ トキ ノ キヨリ= 21414.78 m
k ? 30 ----- 30 を入力の場合
カク 30 トノ トキ ノ キヨリ= 6540.23 m
k ? 70 ----- 70 を入力の場合
カク 70 トノ トキ ノ キヨリ= 1374.356 m

```

〔演習問題〕

- (81) 【例題 72】の図の距離 l を入力して、角 k 度を求めるプログラムをつくれ。
- (82) a, b を入力して、 $\sqrt{a^2+b^2}$ を求めるプログラムをつくれ。
- (83) 文字列を $"$ で囲む式を定義し、文字列を入力して、 $"$ で囲んで表示するプログラムをつくれ。
- (84) 文字列から右端 2 文字をとり出した文字列をつくる式を定義し、文字列を入力して、右端 2 文字を表示するプログラムをつくれ。
- (85) 文字列から左端 2 文字と右端 2 文字をとり出してつなげる式を定義し、次に文字列を入力して前 2 文字と後 2 文字をつないで表示するプログラムをつくれ。
- (86) "Microsoft" の a 番目から b 個の文字をとり出す式を定義し、次に数 x, y を入力して x 番目から y 個の文字をとり出して表示するプログラムをつくれ。

〔解 答〕

(81) 【プログラム例】

```

DEF fna (l) = ATN(3776 / l) / 3.14159 * 180 ----- k 度を求める式の定義
INPUT "l "; l ----- l の入力
k = fna(l) ----- k 度を求める
PRINT "キヨリ"; l; "m ノ トキ ノ カクト="; k; "ト" ----- 結果の表示
END

```

【結 果】

```

l ? 21414 ----- 21414 を入力の場合
キヨリ 21414 m ノ トキ ノ カクト= 10.00036 ト
l ? 100000 ----- 100000 を入力の場合
キヨリ 100000 m ノ トキ ノ カクト= 2.162463 ト

```

(82) 【プログラム例】

```

DEF fnx (a, b) = SQR(a * a + b * b) -----  $\sqrt{a^2+b^2}$  を求める式の定義
INPUT "a, b "; a, b ----- a, b の入力
y = fnx(a, b) -----  $\sqrt{a^2+b^2}$  を求める
PRINT ; y ----- 結果の表示
END

```

【結 果】

a, b ? 3, 4 ————— 3, 4 を入力の場合
 5
 a, b ? 25, 19 ————— 25, 19 を入力の場合
 31.40064

(83) 【プログラム例】

DEF fnx\$ (a\$) = CHR\$(34) + a\$ + CHR\$(34) ——— “文字列”の式の定義
 INPUT x\$ ————— 文字列の入力
 y\$ = fnx\$(x\$) ————— 文字列を””で囲む
 PRINT y\$ ————— 結果の表示
 END

【結 果】

? Quick Basic ————— Quick Basic を入力の場合
 “Quick Basic”
 ? タノシイ プログラム ————— タノシイ プログラムを入力の場合
 “タノシイ プログラム”

(84) 【プログラム例】

DEF fnx\$ (a\$) = RIGHT\$(a\$, 2) ——— 右端 2 文字をとり出す式の定義
 INPUT x\$ ————— 文字列の入力
 y\$ = fnx\$(x\$) ————— 文字列から 2 文字をとり出す
 PRINT y\$ ————— 結果の表示
 END

【結 果】

? picture ————— picture を入力の場合
 re
 ? computer ————— computer を入力の場合
 er

(85) 【プログラム例】

DEF fnx\$ (a\$) = LEFT\$(a\$, 2) + RIGHT\$(a\$, 2) ——— 左 2 文字と右 2 文字をとり出す式の定義
 INPUT x\$ ————— 文字列の入力
 y\$ = fnx\$(x\$) ————— 前 2 文字と後 2 文字をとり出してつなぐ
 PRINT y\$ ————— 結果の表示
 END

【結 果】

? giant ————— giant を入力の場合
 gint
 ? teacher ————— teacher を入力の場合
 teer

(86) 【プログラム例】

```
DEF fnx$ (a, b) = MID$("Microsoft", a, b)——"Microsoft"の前から a 番目から b 個とり出す式の定義
INPUT x, y——x, y の入力
PRINT fnx$(x, y)——"Microsoft"の前から x 番目から y 個とり出して表示
END
```

【結 果】

```
? 2,3——2, 3 を入力の例
i cr
? 3,2——3, 2 を入力の例
cr
? 4,5——4, 5 を入力の例
ros of
```

11 章 プロシジャ・サブルーチン

DECLARE SUB, SUB, END SUB, CALL

【例題 73】 プロシジャ・サブルーチン

メインプログラムで"Quick"と表示し、サブルーチン sa で" BASIC"と続けて表示するプログラムをつくれ。

【解 説】 () プロシジャのうちサブルーチンを使います。

- ① サブルーチンはメインプログラムとは別に作成して、メインプログラムから呼び出して何回でも使える便利な手法です。次の例のように、処理 1 を何回も使うとき、毎回その処理を記入せずに、サブルーチンとして別の場所にしまっておいて、呼び出して使えば、処理 1 は 1 回書けばよいことになります。

処理 1 を必要のつど記入したプログラム

```
/
[ 処理 1 ]
/
[ 処理 1 ]
/
[ 処理 1 ]
/
```

(処理 1 の内容が 100 行になるような場合を考えるとこのプログラムの大変さがよくわかります)

処理 1 をサブルーチン化したプログラム

```
/
サブルーチンの呼び出し
/
サブルーチンの呼び出し
/
サブルーチンの呼び出し
/
```

サブルーチン

```
[ 処理 1 ]
```

- ② 最初は簡単な例です。まずサブプログラムに名前をつけて宣言します。

```
DECLARE SUB sa()
```


DECLARE SUB サブプログラム名となります。サブプログラム名の後は()をつけます。この意味は後で示しますが、サブプログラムへ引数を渡すことがあるからです。今回は引数はありません。

- ③ サブプログラムの呼び出しは CALL サブプログラム名またはサブプログラム名のみです。ただし、サブプログラム名のみでサブプログラムを呼び出すときは必ず DECLARE SUB を使ってサブプログラムの宣言をしておかねばなりません。CALL を使う場合は DECLARE SUB がなくても使えます。

- ④ サブプログラムは先頭に SUB サブプログラム名をつけてつくります。SUB プログラム名を入力す

ると（今回は SUB sa ）次のように表示されますので、サブプログラムの内容をかきます。


```
SUB sa
END SUB
```

- ⑤ 本例ではメインプログラムで“Quick”を表示した後、サブプログラム sa を呼んで“BASIC”を表示します。

```
DECLARE SUB sa()      サブプログラム sa の宣言
/
sa
/
END
SUB sa ←              サブプログラム sa
/                     を呼び出して実行
END SUB               終了すると sa の次へもどる
```

【プログラム例】

```
DECLARE SUB sa () —— サブルーチン sa の宣言
PRINT "Quick"; —— "Quick" の表示
sa
END —— サブルーチン sa の呼び出し

SUB sa —— サブルーチン sa
PRINT " BASIC" —— "BASIC" の表示
END SUB ——
```

【結果】

Quick BASIC

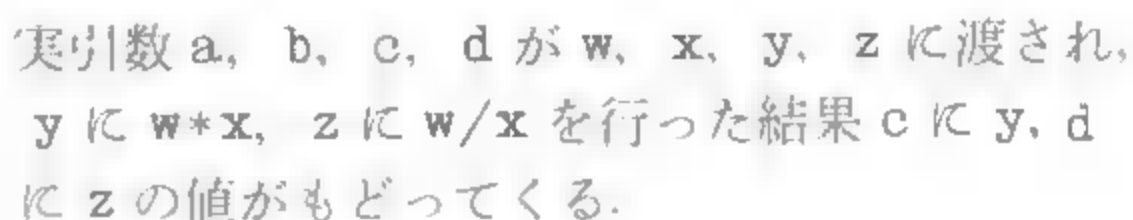
【例題 74】 数値を引数として、サブルーチンの実行

a を 88, b を 22, c と d を 0 としてサブルーチン sc を実行し、サブルーチンで a と b の積と商を求めて、メインプログラムへもどり結果を表示するプログラムをつくれ。

【解説】 ◎ 数値を引数としてサブルーチンを実行します。

- ① DECLARE SUB sc(a,b,c,d) はサブルーチン sc を宣言します。このとき、引数として a, b, c, d を持ち、その値をサブルーチンへ渡します。a, b, c, d は引数が 4 個あり、各々が実数型であることを示します。ここでは、数と型が意味を持ちます。名前は意味を持ちません。サブルーチンに引数を渡せるということは従来の BASIC にはないすばらしい特長です。そのよさについては項⑥で説明します。
- ② サブルーチンでは SUB sc(w,x,y,z) とします。引数を w, x, y, z の順に受けとります。この w, x, y, z を仮引数といいます。仮引数の数と型、この場合は数が 4 個で実数型がサブルーチン呼び出し時の引数すなわち実引数の数と型に合っていなければなりません。

- $a=88, b=22, c=0, d=0$ とする.



- $t = 0$

 $x=5$

wa t.x

/

END

SUB wa(p,q) t=0 を p に, x=5 を q に渡す

FOR i=1 TO q p に $1+2+\dots+q$ の合計を求める

$$p = p + i$$

NEXT i

END SUB

(従来の BASIC)

 $T=0$
$$X=5$$

GO SUB ○○○○

/

```
○○○○ FOR I=1 TO X      T=0,X=5 として
      T=T+I              Tに1+2+...Xを求める
      NEXT I
      RETURN
```

メインプログラムの T, X はサブルーチンの T, X と共通。
メインプログラムで T, X を使うとサブルーチンを使わなくてもサブルーチンの T, X の値も代入されている。

【プログラム例】

```
DECLARE SUB sc (a, b, c, d) —— サブルーチン sc の宣言
a = 88: b = 22: c = 0: d = 0 —— a, b, c, d に初期値を代入
sc a, b, c, d —— サブルーチン sc の呼び出し
PRINT a; "*"; b; "="; c —— 結果の表示
PRINT a; "/"; b; "="; d ——
END
```

```
SUB sc (w, x, y, z) —— サブルーチン sc
  y = w * x —— y と z に計算結果を代入
  z = w / x ——
END SUB ——
```

【結果】

```
88 * 22 = 1936
88 / 22 = 4
```

【例題 75】 文字列を引数とするサブルーチン

メインプログラムはサブルーチン sa1 と sa2 の宣言と各々の呼び出しとし、サブルーチン sa1 で "Let's study", サブルーチン sa2 で "[Quick BASIC.] を表示するプログラムをつくれ。ただし、サブルーチン sa1 の呼び出しで引数として "Let's study", サブルーチン sa2 の呼び出しで引数として "[Quick BASIC." を持つものとする。

【解説】 ◎ 文字列を引数としてサブルーチンを呼び出し実行します。

- ① DECLARE SUB sa1(a\$)ではサブルーチン sa1 を宣言します。このとき(a\$)によって引数が文字型であることを明示します。
- ② sa1 "Let's study"では実引数として "Let's study" をサブルーチンへ渡します。
- ③ サブルーチン sa1 は SUB sa(x\$)で始まります。仮引数が x\$ です。実引数の文字列を x\$ に受けとります。

a\$が文字型を示す

```

DECLARE SUB sa1(a$)      サブルーチンの宣言
/
sa1 "○○○○○"           サブルーチンの呼び出し
/
END                      引き渡す
SUB sa1(x$)              仮引数は実引数と同じ数と型にする
                          サブルーチン
/
END SUB

```

【プログラム例】

```

DECLARE SUB sa1(a$)      サブルーチン sa1 の宣言
DECLARE SUB sa2(b$)      サブルーチン sa2 の宣言
sa1 "Let's study"        サブルーチン sa1 の呼び出し
sa2 "Quick BASIC."       サブルーチン sa2 の呼び出し
END

```

```

SUB sa1(x$)              サブルーチン sa1
  PRINT x$;              x$の表示
END SUB

```

```

SUB sa2(y$)              サブルーチン sa2
  PRINT y$               y$の表示
END SUB

```

【結果】

Let's study Quick BASIC.

【例題 76】 CALL によるサブルーチンの呼び出し

x%を12、y%を25、z\$を"="としてサブルーチン abc を CALL で呼び出し、x%とy%の積を求め式を含めて結果を表示するプログラムをつくれ。

【解説】 ◎ CALL でサブルーチンを呼び出します。

- ① CALL サブルーチン (引数) によって、引数を持ってサブルーチンを呼び出します。CALL を使う場合はサブルーチンの宣言はしていなくてもサブルーチンを呼び出せます。

【プログラム例】

```

DECLARE SUB abc(x%, y%, z$)  サブルーチン abc の宣言
x% = 12: y% = 25: z$ = "="  x, y, z$の代入
CALL abc(x%, y%, z$)        サブルーチンの呼び出し
END

```

```

SUB abc(p%, q%, r$)          サブルーチン
  PRINT p%; "*" ; q%; r$; p% * q%  積を求めて表示
END SUB

```


【結 果】

12 * 25 = 300

【例題 77】 CALL によるサブルーチンの呼び出しと値の受け渡し

【例題 76】と同様に x% を 12, y% を 25, z\$ を "=" としてサブルーチン abc を call で呼び出し, x% と y% の積を求め, 式を含めて結果を表示した後, $x\%^2$ を求めて, メインルーチンへもどり結果を表示するプログラムをつくれ.

【解 説】 ㉞ サブルーチンからメインルーチンへの値の渡し方を学びます.

- ① x% を 12, y% を 25, z\$ を "=" としてサブルーチン abc を call して, $x\% * y\%$ を求めて表示する部分は【例題 76】と同じです. ただし, それに加え zz% の引数をつけます.
- ② 次にサブルーチンの中で次のようにします.

d% = aa% * aa%

この結果 aa% は引き渡された x% の値, bb% は引き渡された y% の値, d% は aa% すなわち x% の 2 乗の値, cc\$ は "=" となります. この状態でメインへもどります. 結果はサブルーチンを call した対応する引数へ返されますから次のようになります.

x%	y%	z\$	zz%	
↑	↑	↑	↑	したがって,
aa%	bb%	cc\$	d%	zz% は d% すなわち $aa\%^2$ すなわち $x\%^2$ となる

【プログラム例】

```

DECLARE SUB abc (a%, b%, c$, d%) — サブルーチン abc の宣言
x% = 12 — 値の代入
y% = 25
z$ = "="
CALL abc(x%, y%, z$, zz%) — サブルーチンの呼び出し
PRINT x%; "^2="; zz% — 結果の表示 zz% はサブルーチンで求める
END

SUB abc (aa%, bb%, cc$, d%) — サブルーチン
  PRINT aa%; "*"; bb%; cc$; aa% * bb% — x% * y% を表示
  d% = aa% * aa% — x% の 2 乗を求めて d% へ代入
END SUB — この値がメインルーチンへ返される

```

【結 果】

12 * 25 = 300
12 ^2 = 144

【例題 78】 配列を引数とするサブルーチン

下の 30 データを配列に読み、その配列を実引数としてサブルーチン `abcd` を呼び出し実行して和を求めるプログラムをつくれ。

データ: 2, 3, 5, 8, 4, 6, 1, 3, 8, 4
 2, 5, 6, 4, 2, 8, 9, 7, 2, 5
 3, 4, 4, 2, 7, 1, 9, 8, 8, 2

【解 説】 ◎ 配列を実引数としてサブルーチンを実行します。

- ① `DECLARE SUB abcd(q())`で配列を引数とするサブルーチン `abcd` を宣言します。 `q()` は 1 つの実数型配列が引数であることを示します。ここでは配列の数と型が問題となります。
- ② `CALL abcd(x())`は配列 `x()`を実引数としてサブルーチン `abcd` を呼び出し実行します。
- ③ `SUB abcd(y())`は `y()`が仮引数で実引数 `x()`を仮引数 `y()`が受けとります。したがって、`y(0)`は `x(0)`、`y(1)`は `x(1)`、`y(2)`は `x(2)`…`y(29)`は `x(29)`となります。

【プログラム例】

```

DECLARE SUB abcd (q())——サブルーチン abcd の宣言
DIM x(30)——配列 x(30) に値を代入
FOR i = 0 TO 29
    READ x(i)
NEXT i
CALL abcd(x())——サブルーチン abcd を呼び出し実行
END
DATA 2,3,5,8,4,6,1,3,8,4
DATA 2,5,6,4,2,8,9,7,2,5
DATA 3,4,4,2,7,1,9,8,8,2

SUB abcd (y())——サブルーチン abcd
    wa = 0——和を求める
    FOR i = 0 TO 29
        wa = wa + y(i)
    NEXT i
    PRINT wa——和の表示
END SUB

```

【結 果】

142

【例題 79】 文字配列を引数とするサブルーチン

下のデータを配列に読み、それを実引数としてサブルーチン `abcd` を呼び出し、表示するプログラムをつくれ。

データ: Tokyo, Osaka, Nagoya, Kochi, Miyazaki,
 Sapporo, Sendai, Morioka, Ube, Kokura

【解 説】 ◎ 文字配列を引数としてサブルーチン呼び出します。

- ① DECLARE SUB abcd(x\$())によってサブルーチン abcd を宣言します。引数は文字配列とします。x\$()が文字配列を示します。
- ② CALL abcd(e\$())で実引数を e\$()としてサブルーチン abcd を呼び出します。
- ③ サブルーチン abcd は仮引数 y\$()で実引数 e\$()を受けとります。

【プログラム例】

```

DECLARE SUB abcd (x$())——サブルーチン abcd の宣言
DIM e$(10)——配列 e$()にデータを読み込む
FOR i = 0 TO 9
    READ e$(i)
NEXT i
CALL abcd(e$())——サブルーチン abcd を呼び出す
END
DATA Tokyo, Osaka, Nagoya, Kochi, Miyazaki
DATA Sapporo, Sendai, Morioka, Ube, Kokura

SUB abcd (y$())——サブルーチン abcd
    FOR i = 0 TO 9——表示
        PRINT y$(i)
    NEXT i
END SUB

```

【結 果】

```

Tokyo
Osaka
Nagoya
Kochi
Miyazaki
Sapporo
Sendai
Morioka
Ube
Kokura

```

【例題 80】 引数の型宣言の仕方

整数型、文字型、実数型を引数とするサブルーチン abcd を宣言し、a%を1000、b\$を”プログラム”、c!を2.56 としてサブルーチン abcd を呼び出し、順に表示するプログラムをつくれ。

【解 説】 ◎ 宣言文の中で引数の型の書き方を示します。

- ① DECLARE SUB abcd(x AS INTEGER, y AS STRING, z AS SINGLE)としてサブルーチン abcd を宣言します。このとき、引数は数と型が問題ですから x AS INTEGER によって整数型が1つ、y AS STRING によって文字型が1つ、z AS SINGLE によって実数型が1つを指定します。これは(x%, y\$, z!)で表わすこともできます。次に各型の表わし方を示します。

型	型の指定の仕方①	型の指定の仕方②
整数型	x%	x AS INTEGER

長整数型	x&	x AS LONG
単精度実数型	x!	x AS SINGLE
倍精度実数型	x#	x AS DOUBLE
文字型	x\$	x AS STRING

【プログラム例】

```

DECLARE SUB abcd (x AS INTEGER, y AS STRING, z AS SINGLE) —— サブルーチンの宣言
a% = 1000: b$ = "プログラム": c! = 2.56 —— 変数に初期値を代入
CALL abcd(a%, b$, c!) —— サブルーチン abcd の呼び出し
END

```

```

SUB abcd (o%, p$, q!) —— サブルーチン abcd
  PRINT o%; p$; q! —— 表示
END SUB

```

【結 果】

1000 プログラム 2.56

【例題 81】 サブルーチンからサブルーチンの呼び出し

メインプログラムで"Quick"を表示し、サブルーチン sa を呼び出して"□BASIC"を表示した後、このサブルーチンからサブルーチン saa を引数を"□オ□マナビマシヨウ"として呼び出し表示するプログラムをつくれ。

【解 説】 ① サブルーチンからサブルーチンを呼び出します。

- ① メインプログラムからサブルーチンを呼び出すのと同じ方法でサブルーチンから別のサブルーチンを呼び出すことができます。
- ② 本例ではメインプログラムで"Quick"を表示した後、引数なしでサブルーチン sa を呼び出し、サブルーチン sa で"□BASIC"を表示し、次に、そこからサブルーチン saa を引数を"□オ□マナビマシヨウ"として呼び出し表示します。

【プログラム例】

```

DECLARE SUB sa () —— サブルーチン sa の宣言
DECLARE SUB saa (a$) —— サブルーチン saa の宣言
PRINT "Quick"; —— "Quick" の表示
sa —— サブルーチン sa の呼び出し
END

```

```

SUB sa —— サブルーチン sa
  PRINT " BASIC"; —— "□BASIC" の表示
  saa " オ マナビ マシヨウ" —— サブルーチン saa の呼び出し
END SUB

```

```

SUB saa (x$) —— サブルーチン saa
  PRINT x$ —— x$ の表示
END SUB

```

【結 果】

Quick BASIC オ マナビマシヨウ

〔演習問題〕

- (87) "プロシージャ ノ ヨビダシ"を表示した後サブルーチン a を呼び出し、サブルーチン a と表示するプログラムをつくれ。
- (88) a を 50, b を 36, c と d を 0 としてそれを実引数としてサブルーチン sb を呼び出し実行して、a と b の和と差を求め、メインプログラムにもどって結果を表示するプログラムをつくれ。
- (89) "Quick BASIC"を表示した後サブルーチン a を呼び出して、"□MS-DOS"を表示するプログラムをつくれ。
- (90) サブルーチン abcd を引数が 2 つの実数型として宣言し、125.357 と 18.2546 を実引数としてサブルーチン abcd を呼び出し、積を求めて表示するプログラムをつくれ。
- (91) 下のデータを使って、配列 e\$(10) と f(10) に読み、これを実引数としてサブルーチン abcd を呼び出し、表示するプログラムをつくれ。
- データ:Tokyo, 10, Osaka, 8, Nagoya, 6, Kochi,
3, Miyazaki, 4, Sapporo, 5, Sendai, 4,
Morioka, 3, Ube, 2, Kokura, 4
- (92) メインプログラムで aa\$を"□BASIC", bb\$を"□C", cc\$を"□Fortran"として、サブルーチン a, b, c を呼び出し、"Quick"の後に各々aa\$, bb\$, cc\$の内容を追加して表示するプログラムをつくれ。
- (93) 次の配列データをサブルーチンに渡し、番号を入力して、その順番のデータを表示するプログラムをつくれ。
- Ikebukuro, Otsuka, Sugamo, Komagome, Tabata,
Nishinippori, Nippori, Uguisudani, Ueno,
Okachimachi, Akihabara, Kanda, Tokyo
- (94) (93) の配列データをサブルーチンに引き渡し、"T"で始まる駅名を表示するプログラムをつくれ。
- (95) 3.14159 を実引数としてサブルーチン a にわたし、半径 10 の円の面積を求めるプログラムをつくれ。
- (96) 実数 23.857 と整数 3 を実引数としてサブルーチン a を呼び出し、 23.857^3 を求めるプログラムをつくれ。
- (97) 整数 x%, y%を入力して、x%の y%乗をサブルーチンで求めて表示するプログラムをつくれ。ただし、x%の y%乗が 10000 をこえるときはさらにサブルーチンへ分岐して"x%^y%>=10000"と表示するものとする。

〔 解 答 〕

(87) 【プログラム例】

```

DECLARE SUB a ()——サブルーチン a の宣言
PRINT "フ°ロシーシ`ャ ノ ヨヒ`タ`シ"——表示
a——サブルーチン a の呼び出し
END

```

```

SUB a——サブルーチン a
    PRINT "サブ`ルー`チン a"——表示
END SUB

```

【結 果】

```

フ°ロシーシ`ャ ノ ヨヒ`タ`シ
サブ`ルー`チン a

```

(88) 【プログラム例】

```

DECLARE SUB sb (a, b, c, d)——サブルーチン sb の宣言
a = 50: b = 36: c = d = 0——a, b, c, d の初期値を代入
sb a, b, c, d——サブルーチン sb の呼び出し
PRINT a; "+"; b; "="; c——結果の表示
PRINT a; "-"; b; "="; d——結果の表示
END

```

```

SUB sb (w, x, y, z)——サブルーチン sb
    y = w + x——和と差を求める
    z = w - x
END SUB

```

【結 果】

```

50 + 36 = 86
50 - 36 = 14

```

(89) 【プログラム例】

```

DECLARE SUB a ()——サブルーチン a の宣言
PRINT "Quick Basic";——表示
CALL a——サブルーチン a の呼び出し
END

```

```

SUB a——サブルーチン a
    PRINT " MS-DOS"——表示
END SUB

```

【結 果】

```

Quick Basic MS-DOS

```


(90) 【プログラム例】

```

DECLARE SUB abcd (x AS SINGLE, y AS SINGLE) —— サブルーチン abcd の宣言
a! = 125.357: b! = 18.2546 —— a と b に値を代入
CALL abcd(a!, b!) —— サブルーチン abcd の呼び出し
END

```

```

SUB abcd (o!, p!) —— サブルーチン abcd
  PRINT o!; "*"; p!; "="; o! * p! —— 積を表示
END SUB

```

【結 果】

125.357 * 18.2546 = 2288.342

(91) 【プログラム例】

```

DECLARE SUB abcd (x$(), y()) —— サブルーチンの宣言
DIM e$(10), f(10) —— 配列にデータの読み込み
FOR i = 0 TO 9
  READ e$(i), f(i)
NEXT i —— サブルーチン abcd の呼び出し
CALL abcd(e$(), f()) ——
END
DATA Tokyo, 10, Osaka, 8, Nagoya, 6, Kochi, 3, Miyazaki, 4
DATA Sapporo, 5, Sendai, 4, Morioka, 3, Ube, 2, Kokura, 4

```

```

SUB abcd (y$(), z()) —— サブルーチン abcd
  FOR i = 0 TO 9 —— 表示
    PRINT y$(i); z(i)
  NEXT i
END SUB

```

【結 果】

Tokyo 10
 Osaka 8
 Nagoya 6
 Kochi 3
 Miyazaki 4
 Sapporo 5
 Sendai 4
 Morioka 3
 Ube 2
 Kokura 4

(92) 【プログラム例】

```

DECLARE SUB a (z$) —— サブルーチン a, b, c の宣言
DECLARE SUB b (z$)
DECLARE SUB c (z$)
aa$ = " Basic" —— 値の代入
bb$ = " C"
cc$ = " Fortran"

```

```
CALL a(aa$)
CALL b(bb$)
CALL c(cc$)
END
```

サブルーチンの呼び出し

```
SUB a (zz$)
  PRINT "Quick" + zz$
END SUB
```

サブルーチン a

```
SUB b (zz$)
  PRINT "Quick" + zz$
END SUB
```

サブルーチン b

```
SUB c (zz$)
  PRINT "Quick" + zz$
END SUB
```

サブルーチン c

【結 果】

```
Quick Basic
Quick C
Quick Fortran
```

(93) 【プログラム例】

```
DECLARE SUB a (x$())
DIM z$(1 TO 13)
FOR i = 1 TO 13
  READ z$(i)
NEXT i
CALL a(z$())
END
DATA Ikebukuro, Otsuka, Sugamo, Komagome, Tabata
DATA Nishinippori, Nippori, Uguisudani, Ueno
DATA Okachimachi, Akihabara, Kanda, Tokyo
```

サブルーチン a の宣言
配列 z\$ に値を読み込む
配列の値を実引数としてサブルーチン a を呼び出す

```
SUB a (zz$())
  INPUT x%
  PRINT zz$(x%)
END SUB
```

サブルーチン a
x% を入力して添字を x% とする配列の値を表示

【結 果】

```
? 3          3 を入力の例
Sugamo
? 7          7 を入力の例
Nippori
```

(94) 【プログラム例】

```
DECLARE SUB a (x$())
DIM z$(1 TO 13)
FOR i = 1 TO 13
  READ z$(i)
NEXT i
```

サブルーチン a の宣言
配列 z\$ に値を読み込む

```
CALL a(z$()) ————— 配列の値を実引数としてサブルーチン a を呼び出す
END
DATA Ikebukuro, Otsuka, Sugamo, Komagome, Tabata
DATA Nishinippori, Nippori, Uguisudani, Ueno
DATA Okachimachi, Akihabara, Kanda, Tokyo
```

```
SUB a (zz$()) ————— サブルーチン a
  FOR i = 1 TO 13 ————— "T"で始まる文字列を表示
    IF ASC(zz$(i)) = 84 THEN PRINT zz$(i)
  NEXT i
END SUB
```

【結 果】

```
Tabata
Tokyo
```

(95) 【プログラム例】

```
DECLARE SUB a (x!) ————— サブルーチン a を宣言
z! = 3.14159 ————— z!に値を代入
CALL a(z!) ————— z!を実引数としてサブルーチン a を呼び出す
END
```

```
SUB a (zz!) ————— サブルーチン a
  s! = 10 * 10 * zz! ————— s!に半径10の円の面積を求めて表示
  PRINT s!
END SUB
```

【結 果】

```
314.159
```

(96) 【プログラム例】

```
DECLARE SUB a (x!, y%) ————— サブルーチン a の宣言
z! = 23.857 ————— z!, zz%に値を代入
zz% = 3 —————
CALL a(z!, zz%) ————— サブルーチン a を呼び出す
END
```

```
SUB a (qq!, pp%) ————— サブルーチン a
  PRINT qq! ^ pp% ————— z!の zz%累乗を求めて表示
END SUB
```

【結 果】

```
13578.37
```

(97) 【プログラム例】

```

DECLARE SUB b (r%, s%) ----- サブルーチン a, b の宣言
DECLARE SUB a (xx%, yy%) -----
INPUT x%, y% ----- x%, y% の入力
CALL a(x%, y%) ----- サブルーチン a を呼び出す
END

SUB a (qq%, pp%) ----- サブルーチン a
  IF qq% ^ pp% <= 10000 THEN
    PRINT qq% ^ pp%
  ELSE
    CALL b(qq%, pp%)
  END IF
END SUB -----

SUB b (r%, s%) ----- サブルーチン b
  PRINT r%; "^"; s%; ">=10000"
END SUB -----

```

【結 果】

? 23,5	23, 5 を入力の例
23 ^ 5 >=10000	
? 3,2	3, 2 を入力の例
9	
? 3,5	3, 5 を入力の例
243	

12章 プロシジャ・ファンクション

DECLARE FUNCTION, FUNCTION, END FUNCTION

【例題 82】 プロシジャ・ファンクション

数 x を入力して x の階乗($x!$) ($x!$ は単精度実数型を示すのではなく x の階乗を示す数学記号である) を求めるプログラムをつくれ。ただし、階乗を求める部分に FUNCTION を使うものとする。

【解 説】 ◎ FUNCTION を使って階乗を求めます。

- ① FUNCTION はサブルーチン同様の考えで使います。SUB との違いは FUNCTION では自身が値を持つことです。具体的にみていきましょう。
- ② DECLARE FUNCTION Kaijo(y AS DOUBLE) では FUNCTION として Kaijo を宣言します。このとき引数が DOUBLE 型であることを示します。これは SUB の宣言の仕方と同様です。
- ③ FUNCTION Kaijo(a#) ~ END FUNCTION は SUB 名前 (仮引数) ~ END SUB と同様です。Kaijo という名前の FUNCTION を示し、仮引数として a# を持っています。実引数を a# に受けとります。

本例の処理の内容は kaijo に階乗の結果を持ちます。次に a# が 0 のとき Kaijo に 0, a# が 0 でないとき Kaijo に j# を代入します。すなわち、Kaijo 自身が値を持つことが SUB と違います。SUB では引数どうして値をやりとりしましたが、SUB の名前は値を持っていませんでした。

- ④ メインプログラムの Kaijo(x#) により x# を引数として、FUNCTION Kaijo を呼び出し実行します。FUNCTION では階乗を求めて Kaijo に代入していますので Kaijo(x#) は $x!$ (x の階乗) となります。SUB と違って Kaijo(x#) 自身が値を持っています。

【プログラム例】

```

DECLARE FUNCTION kaijo (y AS DOUBLE) ----- ファンクション kaijo の宣言
INPUT x# ----- x# の入力
PRINT x#; "!"; "="; kaijo(x#) ----- kaijo を呼び出し実行し結果を表示
END

FUNCTION kaijo (a#) ----- ファンクション kaijo
  j# = 1 ----- 階乗を j# に求める
  FOR i = a# TO 2 STEP -1
    j# = j# * i
  NEXT i
  IF a# = 0 THEN kaijo = 0 ELSE kaijo = j# ----- ファンクション kaijo に階乗の結果を代入
END FUNCTION -----

```

【結 果】

```
? 11          11 を入力の場合
11 != 3.99168E+07

? 9           9 を入力の場合
9 != 362880

? 4           4 を入力の場合
4 != 24
```

【例題 83】 FUNCTION に引数を 2 つ渡す

数 x , y を入力して、次にその値をファンクション `seki` に与えて、積を求め、結果をメインプログラムに返して表示するプログラムをつくれ。

【解 説】 ◎ FUNCTION に引数を 2 つ渡します。

① DECLARE FUNCTION `seki(x,y)` で、ファンクション `seki` を宣言します。引数は x と y です。これは浮動小数点型の 2 つの値です。ここでは変数名 x , y が問題ではなく、数 (2 個) と型が問題です。

② ファンクション `seki` は次のとおりです。

```
FUNCTION seki(a,b)
seki=a*b
END FUNCTION
```

名前は `seki` です。引数は a , b で、 xx の値を a 、 yy の値を b に引きうけます。 $a*b$ の結果がファンクション `seki` の値となります。

③ ファンクションの値は 3 行目の `seki(xx,yy)` で表示します。 xx , yy を実引数としてファンクション `seki` を実行し、その値が `seki(xx,yy)` です。

【プログラム例】

```
DECLARE FUNCTION seki (x, y) ----- ファンクション seki の宣言
INPUT xx, yy ----- xx, yy の入力
PRINT xx; "*"; yy; "="; seki(xx, yy) ----- ファンクション seki(xx,yy) の表示
END

FUNCTION seki (a, b) ----- ファンクション seki
    seki = a * b
END FUNCTION -----
```

【結 果】

```
? 15,3          15 と 3 を入力の場合
15 * 3 = 45

? -25,9          -25 と 9 を入力の場合
-25 * 9 = -225
```


【例題 84】 ファンクションの値でループを終了

文字を入力し, "E" または "e" が入力されたとき全体を順につないで表示するプログラムをつくれ. ただし, "E" または "e" が入力されたときファンクション `handan` の値を 1 として文字の入力を終了し, それ以外では `handan` の値を 0 として, 次の入力を行うものとする.

【解 説】 ◎ **WHILE** 条件の条件をファンクションの値で行います.

- ① 次のループはファンクション `handan` の値が 1 でないときくり返しをします. 1 のときくり返しを終了して `s$` を表示します.

```
WHILE handan(b$) <> 1  —— ファンクション handan を実行して 1 以外なら
    s$ = s$ + b$        —— くり返し, 1 ならループの外へ
WEND                  —— 入力された b$ を順に s$ につないでいく
PRINT s$
```

- ② ファンクション `handan` は文字列 `b$` を入力して "E" または "e" のとき `handan` の値を 1 とし, それ以外では 0 とします. 引数 `b$` はそのままメインプログラムの `b$` に引き渡されます.

【プログラム例】

```
DECLARE FUNCTION handan (a AS STRING) —— ファンクション handan の宣言引数は文字列が 1 つ
s$ = "" —— handan が 1 以外のときくり返し s$ に文字をつないでいく
WHILE handan(b$) <> 1
    s$ = s$ + b$
WEND
PRINT s$ —— s$ の表示
END

FUNCTION handan (b$) —— ファンクション handan
    INPUT b$          —— b$ の入力
    IF b$ = "E" OR b$ = "e" THEN z = 1 ELSE z = 0
    handan = z        —— b$ が "E" または "e" のときファンクション
                        —— の値が 1 その他は 0
END FUNCTION
```

【結 果】

```
? b      b, o, o, k, e を入力の例
? o
? o
? k
? e
book
```

【例題 85】 配列の値をファンクションへ引き渡す

下のデータを元にして、数 x を入力し、最初から x 個の平均を求めて表示するプログラムをつくれ。ただし、メインプログラムで x 個の入力をし、配列に元データをよんで、 x 個とその配列の値をファンクションに引き渡し、ファンクションで平均を求めるものとする。

5, 3, 4, 8, 6, 7, 1, 9, 5, 6

8, 9, 2, 4, 5, 3, 2, 9, 8, 4

9, 7, 5, 8, 4, 2, 2, 9, 2, 5

【解 説】 ◎ 配列をファンクションへ渡します。

① 配列名をかくことで配列をファンクションへ引き渡すことができます。

【プログラム例】

```

DECLARE FUNCTION heikin (x(), y) —— ファンクション heikin の宣言, 配列と整数型を引数とする
DIM a(30) —— 配列の宣言
INPUT xx —— xx% の入力
FOR i = 1 TO xx —— 配列 a% に xx 個のデータを読む
    READ a(i)
NEXT i
h = heikin(a(), xx) —— ファンクション heikin を表示
PRINT xx; "コノハイキヘン="; h
END
DATA 5, 3, 4, 8, 6, 7, 1, 9, 5, 6
DATA 8, 9, 2, 4, 5, 3, 2, 9, 8, 4
DATA 9, 7, 5, 8, 4, 2, 2, 9, 2, 5

```

```

FUNCTION heikin (n(), p) —— ファンクション heikin, 配列を n%(i) に個数を p% にうける
    t = 0 —— 和を求める
    FOR i = 1 TO p
        t = t + n(i)
    NEXT i
    heikin = t / p —— 平均を求める, 平均を heikin の値とする
END FUNCTION

```

【結 果】

? 10 10 コノハイキヘン= 5.4 10 を入力の例

? 16 16 コノハイキヘン= 5.3125 16 を入力の例

? 30 30 コノハイキヘン= 5.366667 30 を入力の例

? 1 1 コノハイキヘン= 5 1 を入力の例

〔演習問題〕

- (98) 数 xx , yy を入力し、これをファンクション wa に引き渡して和を求め結果をメインプログラムにもどして表示するプログラムをつくれ。
- (99) 文字列 $x\$$ を入力し、その文字列をファンクション $mojisu$ に渡しその文字列の文字数をファンクション $mojisu$ の値としてメインプログラムにもどして表示するプログラムをつくれ。
- (100) 整数を入力して、その値が 0 のとき値を 999 とし、それ以外では値を 0 とするファンクション $handan$ をつくり、引数として、入力された値をメインプログラムにもどして和を求め、0 が入力されたとき和を表示するプログラムをつくれ。
- (101) 整数 $x\%$ を入力して、 $3x^2+2x+5$ の値を求めるプログラムをつくれ。ただし、式の値は FUNCTION を使って求めるものとする。
- (102) 整数 $x\%$ を入力して、 $x\%$ 度の \sin の値を求めるプログラムをつくれ。
- (103) 整数 $x\%$ を入力して、 $1\sim x\%$ の和を求めるプログラムをつくれ。

〔解 答〕

(98) 【プログラム例】

```

DECLARE FUNCTION wa (x, y) ————— ファンクション和の宣言
INPUT xx, yy —————  $xx$ ,  $yy$  の入力
PRINT xx; "+"; yy; "="; wa(xx, yy) — ファンクション  $wa$  の表示
END

```

```

FUNCTION wa (a, b) ————— ファンクション  $wa$ 
    wa = a + b
END FUNCTION —————

```

【結 果】

```

? 154,854          154 と 854 を入力の例
154 + 854 = 1008

```

```

? 21,8956          21 と 8956 を入力の例
21 + 8956 = 8977

```

(99) 【プログラム例】

```

DECLARE FUNCTION mojisu (a AS STRING) — ファンクション  $mojisu$  の宣言
INPUT x$ —————  $x\$$  の入力
PRINT mojisu(x$) ————— ファンクション  $mojisu$  の表示
END

```

```

FUNCTION mojisu (b$) ————— ファンクション  $mojisu$ 
    mojisu = LEN(b$) — 文字列の文字数を求める
END FUNCTION —————

```

【結 果】

? moonlight moonlight を入力の場合
9

(100) 【プログラム例】

```

DECLARE FUNCTION handan (a AS INTEGER) — ファンクション handan の宣言
wa = 0 — wa を求めるくり返し
WHILE handan(b%) <> 999 — 入力された値 b%はファンクション
    wa = wa + b% — handan から引き渡される
WEND
PRINT wa — 和の表示
END

```

```

FUNCTION handan (b%) — ファンクション handan
    INPUT b% — b%の入力
    IF b% = 0 THEN handan = 999 ELSE handan = 0 — b%が0 のときファンクションの値を 999,
END FUNCTION — その他のとき 0 とする

```

【結 果】

? 5 5, 7, 9, 7, 2, 4, 5, 0 を入力の場合
? 7
? 9
? 7
? 2
? 4
? 5
? 0
39

(101) 【プログラム例】

```

DECLARE FUNCTION a (a AS INTEGER) — a の定義
INPUT x% — x%の入力
PRINT a(x%) — a を呼び出し結果を表示
END

```

```

FUNCTION a (z%) — ファンクション a
    a = 3 * z% * z% + 2 * z% + 5 —  $3x\%^2 + 2x\% + 5$  を求める
END FUNCTION

```

【結 果】

? 2 2 を入力の場合
21
? 8 8 を入力の場合
213

(102) 【プログラム例】

```

DECLARE FUNCTION a (a AS INTEGER) — a の定義
INPUT x% — x%の入力
PRINT a(x%) — a を呼び出し結果を表示
END

```

```

FUNCTION a (z%)
    a = SIN(z% / 180 * 3.14159)
END FUNCTION

```

ファンクション a
sin (x%度) を求める

【結 果】

```

? 67          67 を入力の場合
.9205045

```

(103) 【プログラム例】

```

DECLARE FUNCTION a (a AS INTEGER)
INPUT x%
PRINT a(x%)
END

```

a の定義
x% の入力
a を呼び出し結果を表示

```

FUNCTION a (z%)
    FOR i% = 1 TO z%
        zz% = zz% + i%
    NEXT i%
    a = zz%
END FUNCTION

```

ファンクション a
1 から x% の和を求める

【結 果】

```

? 5          5 を入力の場合
15
? 100        100 を入力の場合
5050

```

13 章 グローバル変数とローカル変数, プログラムの連結

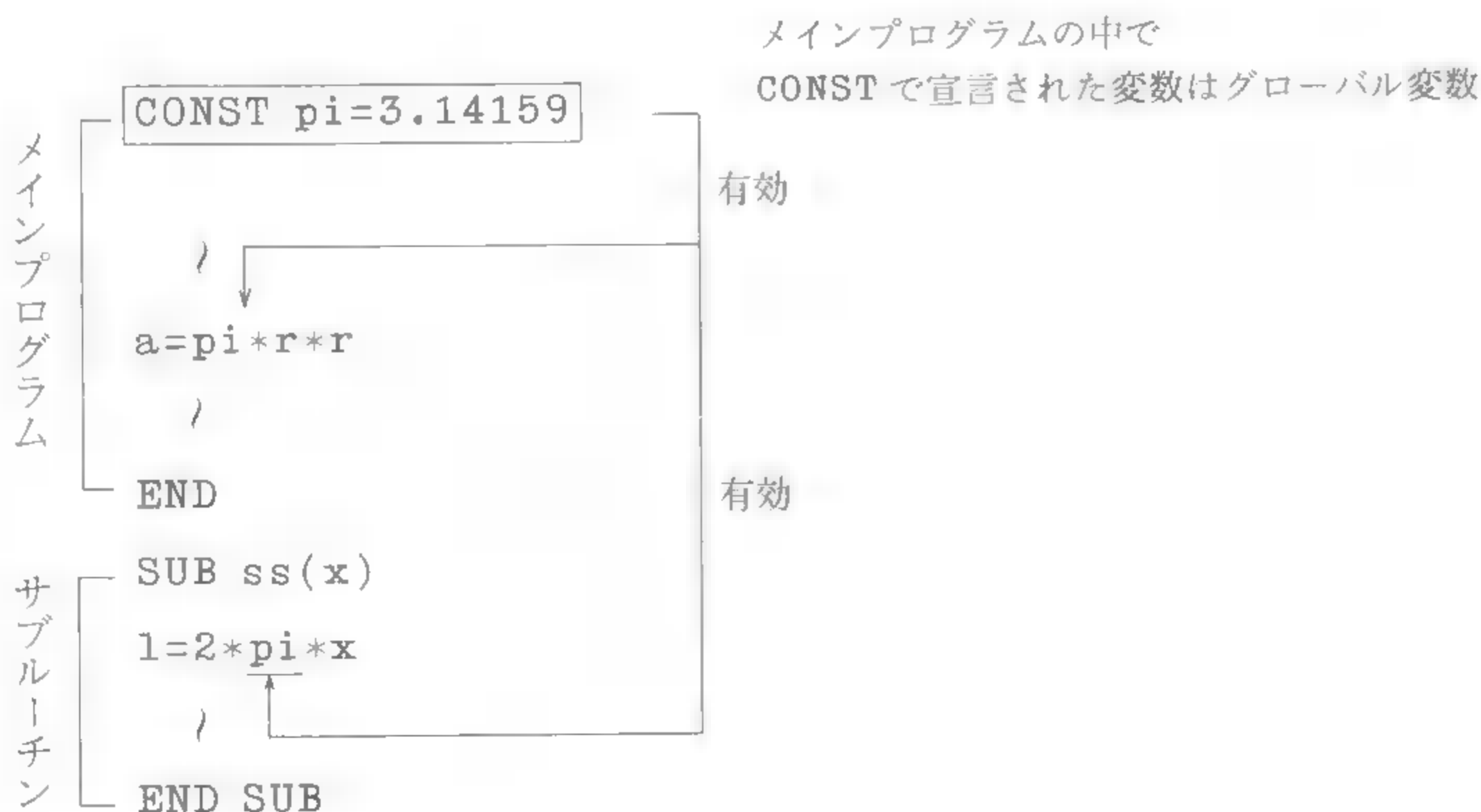
COMMON, COMMON SHARED, CHAIN, STATIC, SHARED

【例題 86】 グローバル変数とローカル変数

変数 π を 3.14159 と CONST で宣言・代入し, 次に半径 r を入力して, 円の面積 (πr^2) を求めて表示し, サブルーチン ss を引数を r として呼び出し, 円周長 ($2\pi \cdot r$) を求めて表示するプログラムをつくれ.

【解 説】 ◎ グローバル変数の有効範囲とローカル変数の有効範囲

- ① Quick BASIC では変数の有効範囲によってグローバル変数とローカル変数に分けます. グローバル変数はその値がすべてのモジュールで有効な変数です. 従来の BASIC では変数はプログラムの中ではすべて有効でしたから, すべての変数がグローバル変数であったともいえます. 一方, 特定のモジュールの中だけで有効な変数をローカル変数といいます. Quick BASIC では特に指定のない限り, 変数はローカル変数です. このことはサブルーチンやファンクションの独立性が高いことを示します.
- ② メインプログラムの中で宣言・代入された定数 $\text{CONST } \pi=3.14159$ はグローバルです. 本例ではメインプログラムの中の π は 3.14159 であり, サブルーチン $ss(x)$ の中の π も 3.14159 が有効です.
- ③ プロシジャの中で宣言される CONST はローカルで, そのプロシジャの中でのみ有効です.



- ④ メインプログラムの中で入力された値を持つ変数 r はローカルです. したがって, メインプログラムの中だけで有効です. したがって, $\text{CALL } ss(r)$ では r の値をサブルーチンへ引数として渡し

ます。なお、メインプログラムの変数 s もローカルです。

- ⑤ サブルーチンでは $2 * \pi * x$ を求めて l に代入し、 x と l を表示します。 π の値はグローバルですから 3.14159 が生きています。 x の値は関数の引数として r の値が引き渡されています。結果は変数 l に代入されます。 l はローカル変数ですからサブルーチンの中でのみ有効です。
- ⑥ サブルーチンの中で x のかわりに r の値を表示させてみます。結果は 0 です。すなわちメインプログラムの r の値はローカルですからサブルーチンの中では 0 です。 r の値はサブルーチンの引数として仮引数 x に渡されているのです。

【参考プログラム例】

```
SUB ss (x)
  l = 2 * pi * x
  PRINT r; l
END SUB
```

x の代わりに r を表示した例

【結 果】

```
? 3.84
3.84 46.32463
0 24.12741
```

3.84 を入力の場合
r は 0 になっていることに注意

【プログラム例】

```
DECLARE SUB ss (x!)
CONST pi = 3.14159
INPUT r
s = pi * r * r
PRINT r; s
CALL ss(r)
END

SUB ss (x)
  l = 2 * pi * x
  PRINT x; l
END SUB
```

サブルーチン ss の宣言
定数の宣言
r の入力
円の面積の計算
結果の表示
サブルーチンを呼び出し、引数を r とする

サブルーチン ss. 仮引数は x
円周長を求める
半径と円周長の表示

【結 果】

```
? 2.5
2.5 19.63494
2.5 15.70795
```

2.5 を入力の場合

【例題 87】 整数型変数と長整数型変数を COMMON で宣言

変数 a, b, c を整数型、 d, e, f を長整数型で COMMON 宣言し、 a, b, c を入力して、 d に積、 e に各々の 2 乗の積、 f に d の 2 乗を求めるプログラムをつくれ。

【解 説】 ◎ 整数型変数と長整数型変数を COMMON で宣言します。

- ① 次の形で COMMON 宣言をします。COMMON で宣言された変数はグローバル変数です。

```
COMMON a, b, c AS INTEGER
```

a, b, c を整数型で宣言

```
COMMON d,e,f AS LONG
```

d, e, f を長整数型で宣言

【プログラム例】

```
COMMON a, b, c AS INTEGER  宣言
COMMON d, e, f AS LONG    宣言
INPUT a, b, c              入力
d = a * b * c              積
e = a * a * b * b * c * c
f = d * d                  積
PRINT a; b; c              表示
PRINT d; e; f              表示
END
```

【結 果】

```
? 12,10,6 12, 10, 6 を入力の例
12 10 6
720 518400 518400
```

【例題 88】 単精度実数型変数と倍精度実数型変数を COMMON で宣言

変数 a, b を単精度実数, c, d を倍精度実数と宣言して, a, b を入力して, 積を c に, 商を d に代入して a, b, c, d を表示するプログラムをつくれ.

【解 説】 ◎ 単精度実数型変数と倍精度実数型変数を COMMON で宣言をします.

① 次の形で COMMON 宣言します. COMMON で宣言された変数はグローバル変数です.

```
COMMON a,b AS SINGLE
```

a, b を単精度実数として宣言

```
COMMON c,d AS DOUBLE
```

c, d を倍精度実数として宣言

【プログラム例】

```
COMMON a, b AS SINGLE  宣言
COMMON c, d AS DOUBLE  宣言
INPUT a, b              入力
c = a * b               積と商
d = a / b               積と商
PRINT a; b              表示
PRINT c; d              表示
END
```

【結 果】

```
? 25.325,12.7896 25.325 と 12.7896 を入力の例
25.325 12.7896
323.8966 1.980124473571777
```

【例題 89】 文字型変数を COMMON で宣言

変数 a を 10 文字の COMMON 宣言をして a に文字列を入力して表示するプログラムをつくれ。

【解 説】 ◎ 文字型変数を COMMON で宣言します。

- ① 次の形で文字変数の文字数を決めて COMMON 宣言をします。COMMON で宣言された変数はグローバル変数です。

```
COMMON a AS STRING*10
```

変数 a を 10 文字列として COMMON 宣言

【プログラム例】

```
COMMON a AS STRING * 10——— 宣言
INPUT a——— 入力
PRINT a——— 表示
END
```

【結 果】

? abcdefg	a.....g Ⓜ の例
abcdefg	
? abcdefghij	a.....j Ⓜ の例
abcdefghijkl	
? abcdefghijklmn	a.....n Ⓜ の例
abcdefghijkl	

【例題 90】 グローバル変数

pi を 3.14159 として CONST で宣言し、次に変数 r をグローバルに宣言してから r に半径を入力して、円の面積を求めて表示した後、サブルーチンを使って、 $2 \times \text{pi} \times r$ (円周長) を求めて表示するプログラムをつくれ。

【解 説】 ◎ プロシジャで有効なグローバル変数を宣言します。

- ① メインプログラムで次のように宣言された変数はグローバルです。モジュール間を通じて共通に使えます。

```
COMMON SHARED 変数
```

- ② メインプログラムで r を入力し $\text{pi} \times r \times r$ を求めた後サブルーチンで $2 \times \text{pi} \times r$ を求めます。今回はサブルーチンへの引数はありません。r の値は入力された値が生きていて半径 r の円周長が求められます。このように r の値はサブルーチンの中でも有効なのです。
- ③ 【例題 86】の r の値、r をサブルーチンへ渡す方法と比べてみてください。従来の BASIC になれた人は r が共通に使える方が便利で簡単と思われるかも知れません。しかし、プログラムが複雑になると、サブルーチンや関数の独立性が高い方が便利になってくるのです。

【プログラム例】

```

DECLARE SUB ss ()——サブルーチン ss の宣言
CONST pi = 3.14159——定数の宣言
COMMON SHARED r——変数 r をグローバルに宣言
INPUT r——r の入力
s = pi * r * r——円の面積の計算
PRINT r; s——半径と面積の表示
CALL ss——サブルーチンの呼び出し、引数はない。
END——半径 r はグローバルなのでサブルーチンでも使える

SUB ss——サブルーチン
  l = 2 * pi * r——円周長の計算
  PRINT r; l——半径と円周長の表示
END SUB

```

【結 果】

```

? 12.5          12.5 を入力の例
12.5    490.8734
12.5    78.53975

```

【例題 91】 ローカル変数

x を 125.33, y を 654.1 として $x*y$ を xy に求めてからサブルーチン ss を呼び出し、x を 54.2, y を 32.58 として $x*y$ を xy に求めて表示し、メインプログラムに戻って、最初の $x*y$ の値を表示するプログラムをつくれ。

【解 説】 ① ローカル変数の働きについて調べます。

- ① メインプログラムで $x=5.33$, $y=654.1$ として $x*y$ を求めて xy に代入します。
- ② 続いてサブルーチンで $x=54.2$, $y=32.58$ として $x*y$ を求めて xy に代入します。結果を表示します。
- ③ メインルーチンに戻って、最初の x と y および xy の値を表示します。x はサブルーチンで 54.2 を代入していますが元の 125.33 が生きています。y も同様に 654.1 が生きています。xy は $54.2*654.1$ です。つまり、変数 x, y, xy はローカル変数のためメインプログラムの中の x, y, xy とサブルーチンの中の x, y, xy ではお互いに独立しているのです。

【プログラム例】

```

DECLARE SUB ss ()——サブルーチン ss の宣言
x = 125.33——x と y に値を代入
y = 654.1——
xy = x * y——xy に x と y の積を代入
CALL ss——サブルーチン ss を呼び出す
PRINT x; "*"; y; "="; xy——x, y, xy の表示
END

SUB ss——サブルーチン ss
  x = 54.2——x と y に値を代入
  y = 32.58——

```

```

xy = x * y ————— xy に x と y の積を代入
PRINT x; "*"; y; "="; xy ——— x, y, xy を表示
END SUB

```

【結 果】

```

54.2 * 32.58 = 1765.836
125.33 * 654.1 = 81978.35

```

【例題 92】 グローバル変数とローカル変数の働き

x をグローバル、 y をローカル変数として、メインプログラムで x と y をまず 0 とした後 10 回のくり返しを行い、 x と y に各々 1 を加え表示します。続いてサブルーチンを呼び出し、 y を 0 とした後 y に 5 を加え x と y を表示するプログラムをつくれ。

【解 説】 ㉓ グローバル変数とローカル変数の働きについて比べながら確認します。

- ① メインプログラムで x をグローバルに宣言します。 x と y を 0 とします。 y は特別に宣言されていませんからローカルです。
- ② 10 回のくり返しをします。 x と y に 1 を加えます。そして表示します。最初は 1, 1 です。サブルーチンへ分岐し、 y を 0 とし、 $y+5$ とした後 x と y を表示します。 x はグローバルですから 1 が生きていて 1、 y はローカルですから 5 となります。したがって、1 回目は 1, 1, 1, 5 です。
- ③ 2 回目は x は 1 に 1 が加えられて 2、メインプログラムの y は元の 1 が生きていて $1+1$ で 2 です。サブルーチンでは x は 2、 y はまず 0 とした後 5 を加えますから 5 です。したがって、2, 2, 2, 5 となります。

【プログラム例】

```

DECLARE SUB ss () ——— サブルーチンの宣言
COMMON SHARED x ——— x をグローバルに宣言
x = 0 ————— x と y を 0 にする
y = 0 —————
FOR i = 1 TO 10 ——— 10 回のくり返し
  x = x + 1 ——— x と y に 1 を加える
  y = y + 1 ———
  PRINT x; y; ——— x と y の表示
  CALL ss ——— サブルーチンの呼び出し
NEXT i
END

SUB ss ————— サブルーチン ss
  y = 0 ————— y を 0 とする
  y = y + 5 ——— y に 5 を加える
  PRINT x; y ——— x と y の表示
END SUB

```

【結 果】

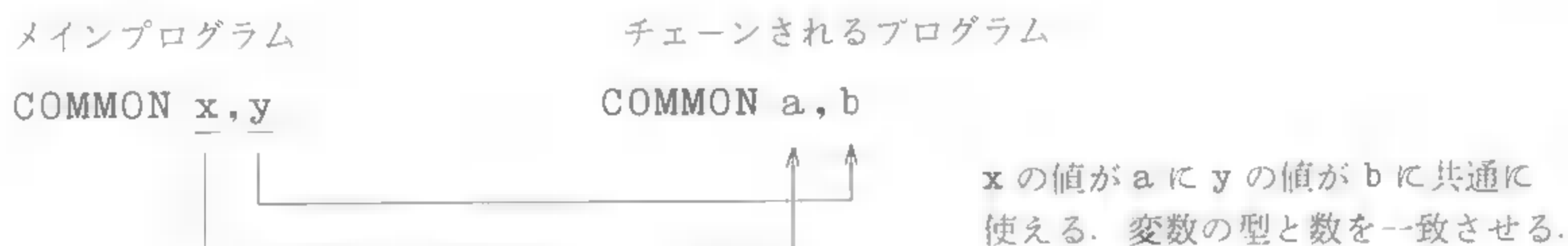
1	1	1	5	———	1 回目
2	2	2	5	———	2 回目
3	3	3	5	———	3 回目
4	4	4	5		
5	5	5	5		
6	6	6	5		
7	7	7	5		
8	8	8	5		
9	9	9	5		
10	10	10	5		

【例題 93】 チェーンしたプログラムで変数を共通に利用

$x=123.45$, y を 258.32 として $x*y$ を求めて表示した後, ree-95.bas のファイルをチェーンして, x と y の値を生かして商を求めて表示するプログラムをつくれ.

【解 説】 ● チェーンしたプログラムで変数の値を共通に使用します.

- ① チェーンしたプログラムで変数を共通に使うためにはメインプログラムとチェーンされるプログラムの先頭で COMMON 宣言をします. この時, 実数の型と数が大切です. 名前は違っていてもかまいません.



- ② 本例では x を 123.45 , y を 258.32 とした後 $x*y$ を求め, ree-95.bas にチェーンします. x の値は a , y の値は b の値として使えます. したがって, a/b は $123.45/258.32$ です.

【プログラム例】

COMMON x, y	変数の宣言
x = 123.45	x, y の値を代入
y = 258.32	
seki = x * y	積を求める
PRINT x; "*"; y; "="; seki	結果の表示
CHAIN "b:ree-95.bas"	b ドライブの ree-95.bas ファイルへチェーン
END	チェーンされるプログラム、ファイル名を "ree-95.bas" として b ドライブに格納しておく

【プログラム例】 (ree-95.bas)

COMMON a, b	変数の宣言
shou = a / b	商の計算
PRINT a; "/" ; b; "="; shou	結果の表示
END	

【結 果】

123.45 * 258.32 = 31889.6
123.45 / 258.32 = .4778956

【例題 94】 ローカル変数を一時的に利用

FUNCTION $x()$ を宣言し、次に $i=65$ から 90 までのくり返しを使って、 i のキャラクタコードの文字を表示した後、FUNCTION で $i+32$ の値を求めそれを FUNCTION の値として、メインプログラムでそれをキャラクタコードとする文字を表示するプログラムをつくれ。

【解 説】 ㉓ ローカル変数の値を一時的に利用します。

- ① i を 65 から 90 までとして、くり返しで $\text{CHR}\$(i)$ を表示します。最初は A 、最後は Z です。
- ② FUNCTION を呼び出します。このとき SHARED i によって、ローカル変数である i を一時的に使えるようにします。したがって、 $x=i+32$ とすると FUNCTION の値は最初は $65+32=97$ となります。メインプログラムに戻って $\text{CHR}\$(x)$ を表示すると a となります。
- ③ 結果は $65, 66, \dots, 90$ をキャラクタコードとする文字 $A, B, C \dots Z$ の各々の後にその値 $+32$ をキャラクタコードとする文字 $a, b, c \dots z$ を表示します。

【プログラム例】

```

DECLARE FUNCTION x ()——FUNCTION の宣言
FOR i = 65 TO 90——i を 65 から 90 までくり返し
    PRINT CHR$(i);——CHR$(i) の表示
    PRINT CHR$(x);——FUNCTION の値をキャラクタコードとする文字の表示
NEXT i
END

FUNCTION x——FUNCTION
    SHARED i——ローカル変数 i を一時的に使う
    x = i + 32——i+32 を FUNCTION の値とする
END FUNCTION

```

【結 果】

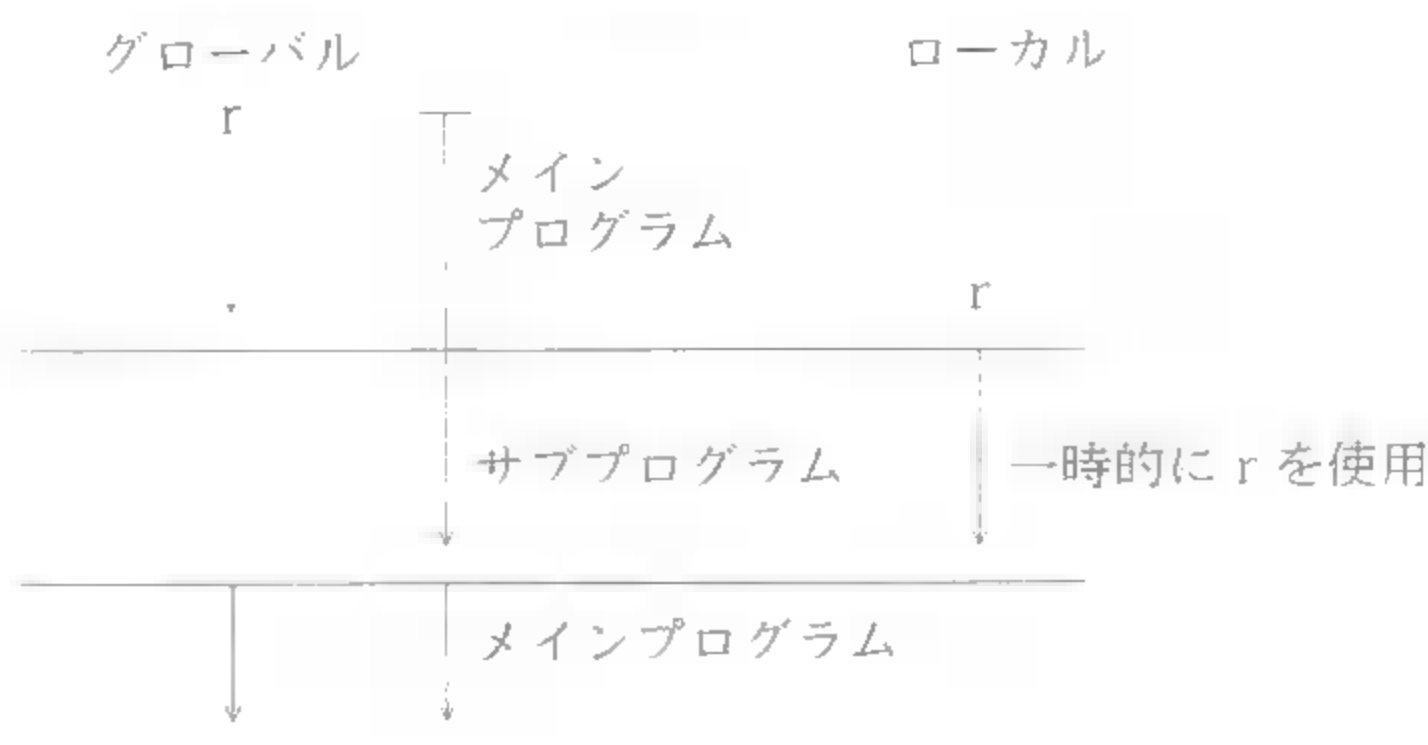
AaBbCcDdEeFfGgHhIiJjKkLlMmNnOoPpQqRrSsTtUuVvWwXxYyZz

【例題 95】 グローバル変数を一時にローカルで使用

グローバルで定数 π を 3.14159 、変数 r を宣言し、 r に値を入力した後サブルーチンを呼び出し、一時的に r をローカル変数として使用し、 r を入力して、円周長を求めて表示した後、メインプログラムへもどって元の r を使って円の面積を求めて表示するプログラムをつくれ。

【解 説】 ㉓ グローバル変数を一時的にローカルに使います。

- ① メインプログラムで r に値を入力した後、サブルーチンを呼び出します。
- ② サブルーチンでは STATIC r と宣言することで r を一時的にサブルーチンの中でのみ使うローカル変数とします。そこで別の r を入力して計算した後メインプログラムへもどります。 r の値は元の r の値が残っています。



【プログラム例】

```

DECLARE SUB ss () —— サブルーチンの宣言
CONST pi = 3.14159 —— 定数 pi の宣言
COMMON SHARED r —— r をグローバルに宣言
INPUT r —— r の入力
CALL ss —— サブルーチン ss の呼び出し
s = pi * r * r —— 円の面積を求める
PRINT r; s —— r と s の表示
END

SUB ss —— サブルーチン
  STATIC r —— グローバル変数 r を一時的にローカルとして使用宣言
  INPUT r —— r の入力
  l = 2 * pi * r —— 円周長を求める
  PRINT r; l —— r と l の表示
END SUB

```

【結 果】

```

? 1.5          1.5 と 2.2 を入力の例
? 2.2
2.2  13.823
1.5  7.068578

```

【例題 96】 配列変数

配列 a(1 TO 10) を宣言して下のデータを読み、次にサブルーチンを呼び出し、配列のデータを共用して表示するプログラムをつくれ。

データ; 1, 5, 6, 2, 7, 8, 3, 6, 5, 4

【解 説】 ◎ 配列変数の有効範囲を調べます。

- ① 配列変数の有効範囲も、普通の変数と同じです。
- ② 本例では DIM a(1 TO 10) により配列変数 a をローカルに配列宣言します。
- ③ サブルーチンでは SHARED a() によって配列変数 a() を共有します。

【プログラム例】

```

DECLARE SUB ss () —— サブルーチンの宣言
DIM a(1 TO 10) —— 配列変数 a の宣言

```

```

FOR i = 1 TO 10 ————— 配列ヘデータを読み込む
    READ a(i)
NEXT i
CALL ss ————— サブルーチンの呼び出し
END
DATA 1, 5, 6, 2, 7, 8, 3, 6, 5, 4

SUB ss ————— サブルーチン
    SHARED a() ————— 配列 a() を共用
    FOR i = 1 TO 10 ————— a(1) ~ a(10) を表示
        PRINT a(i);
    NEXT i
END SUB

```

【結 果】

1 5 6 2 7 8 3 6 5 4

〔演習問題〕

- (104) 変数 wa, sa, x, y をグローバルに宣言して x と y を入力した後 wa に $x+y$, sa に $x-y$ を求めてからサブルーチンに分岐して結果を表示するプログラムをつくれ.
- (105) 変数 x をグローバルに宣言し、数 x を入力してサブルーチンを呼び出し x^3 を求めて表示するプログラムをつくれ.
- (106) 変数 shou%, joyo%, x%, y% をグローバルに宣言して、x%, y% を入力して、商を shou%, 剰余を joyo% に求めてからサブルーチンを呼び出し結果を表示するプログラムをつくれ.
- (107) x% を 54, y% を 25 として、積を求めて表示した後、B ドライブのファイル名 ree-105.bas へチェーンして、x% と y% の値を共通に使って $x\% \div y\%$ の剰余を求めて表示するプログラムをつくれ.
- (108) 変数 x, y をグローバルに宣言し、x, y を入力してからサブルーチンを呼び出し、x, y を一時的にローカルで使用して x, y を入力し、 x^2+y^2 を求めて表示した後、メインプログラムで、元の $x*y$ を求めて表示するプログラムをつくれ.
- (109) 下のデータを配列 a\$(1 TO 10) に読み、サブルーチンを呼んで配列 a\$(1 TO 10) を共用して表示するプログラムをつくれ.
- データ;Tokyo, Osaka, Nagoya, Fukui, Sapporo,
Yokohama, Kawasaki, Takamatsu, Fukuoka, Sendai
- (110) i を 1 から 10 までとしてくり返しを行い、 $i*i=$ の後に i^2 を表示するプログラムをつくれ. ただし、メインプログラムのローカル変数 i を FUNCTION x で一時的に共用して i^2 を求めそれを FUNCTION の値として使うものとする.
- (111) 変数 a を整数型, b を長整数型で COMMON 宣言し、a を入力して a^3 を b に代入して表示するプログラムをつくれ.
- (112) a を単精度実数型, b を単精度実数型, c を倍精度実数型として COMMON 宣言し、a, b を

入力して商を c に求め表示するプログラムをつくれ。

〔 解 答 〕

(104) 【プログラム例】

```

DECLARE SUB ss () —— サブルーチンの宣言
COMMON SHARED wa, sa, x, y —— wa, sa, x, y をグローバルに宣言
INPUT x, y —— x と y の入力
wa = x + y —— 和と差の計算
sa = x - y ——
CALL ss —— サブルーチン ss の呼び出し
END

SUB ss —— サブルーチン ss
  PRINT x; "+"; y; "="; wa —— 和と差の表示
  PRINT x; "-"; y; "="; sa ——
END SUB

```

【結 果】

```

? 5,3          5, 3 を入力の例
5 + 3 = 8
5 - 3 = 2

```

(105) 【プログラム例】

```

DECLARE SUB ss () —— サブルーチン ss の宣言
COMMON SHARED x —— x をグローバルに宣言
INPUT x —— x の入力
CALL ss —— サブルーチン ss を呼び出す
END

SUB ss —— サブルーチン ss
  s = x * x * x —— x3 を s に代入
  PRINT x; s —— x と s を表示
END SUB

```

【結 果】

```

? 12.54          12.54 を入力の例
12.54  1971.935

```

(106) 【プログラム例】

```

DECLARE SUB ss () —— サブルーチンの宣言
COMMON SHARED shou%, joyo%, x%, y% —— グローバルに変数を宣言
INPUT x%, y% —— x%, y% の入力
shou% = x% \ y% —— 商と剰余を求める
joyo% = x% MOD y% ——
CALL ss —— サブルーチン を呼び出す
END

```

```
SUB ss ----- サブルーチン
  PRINT x%; "/"; y%; "="; shou%; "アマリ "; joyo% ----- 結果の表示
END SUB
```

【結 果】

```
? 125,51
125 / 51 = 2 アマリ 23
```

125 と 51 を入力の場合

(107) 【プログラム例】

```
COMMON x%, y% ----- x%, y%の宣言
x% = 54 ----- x%, y%に値を代入
y% = 25 -----
seki% = x% * y% ----- 積を seki%に代入
PRINT x%; "*"; y%; "="; seki% ----- 結果の表示
CHAIN "b:ree-105.bas" ----- ファイル ree-105 へチェーン
END
```

【プログラム例】 (ree-105 のプログラム)

```
COMMON a%, b% ----- a%, b%の宣言
amari% = a% MOD b% ----- 剰余の計算
PRINT a%; "/"; b%; "ノ アマリ="; amari% ----- 結果の表示
END
```

【結 果】

```
54 * 25 = 1350
54 / 25 ノ アマリ= 4
```

(108) 【プログラム例】

```
DECLARE SUB ss () ----- サブルーチンの宣言
COMMON SHARED x, y ----- x, y をグローバルに宣言
INPUT x, y ----- x, y の入力
CALL ss ----- サブルーチンを呼び出す
s = x * y ----- x*y の値の表示
PRINT x; "*"; y; "="; s -----
END
```

```
SUB ss ----- サブルーチン
  STATIC x, y ----- グローバル変数 x, y を一時的にローカルで使用
  INPUT x, y ----- x, y を入力
  s = x * x + y * y ----- x2+y2 を求めて表示
  PRINT x; "^2+"; y; "^2="; s -----
END SUB
```

【結 果】

```
? 12.3,2.5
? 1.35,8.23
1.35 ^2+ 8.23 ^2= 69.5554
12.3 * 2.5 = 30.75
```

12.3, 2.5, 1.35, 8.23 を入力の場合

(109) 【プログラム例】

```

DECLARE SUB ss () —— サブルーチンの宣言
DIM a$(1 TO 10) —— 配列の宣言
FOR i = 1 TO 10 —— 配列にデータを読み込む
    READ a$(i)
NEXT i
CALL ss —— サブルーチンを呼び出して実行
END
DATA Tokyo, Osaka, Nagoya, Fukui, Sapporo
DATA Yokohama, Kawasaki, Takamatsu, Fukuoka, Sendai

```

```

SUB ss —— サブルーチン
    SHARED a$() —— 配列 a$ を共用する
    FOR i = 1 TO 10 —— a$ の表示
        PRINT a$(i)
    NEXT i
END SUB

```

【結 果】

```

Tokyo
Osaka
Nagoya
Fukui
Sapporo
Yokohama
Kawasaki
Takamatsu
Fukuoka
Sendai

```

(110) 【プログラム例】

```

DECLARE FUNCTION x () —— FUNCTION の宣言
FOR i = 1 TO 10 —— i を 1 から 10 までくり返し
    PRINT i; "*"; i; "="; —— i × i = の後に i2 を表示
    PRINT x
NEXT i
END

```

```

FUNCTION x —— FUNCTION x
    SHARED i —— ローカル変数 i を一時的に共用で使う
    x = i * i —— i2 を FUNCTION の値とする
END FUNCTION

```

【結 果】

```

1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
4 * 4 = 16
5 * 5 = 25
6 * 6 = 36
7 * 7 = 49
8 * 8 = 64
9 * 9 = 81
10 * 10 = 100

```


(111) 【プログラム例】

```
COMMON a AS INTEGER
COMMON b AS LONG
INPUT a
b = a * a * a
PRINT a
PRINT b
END
```

【結 果】

```
? 8
8
512
```

(112) 【プログラム例】

```
COMMON a AS SINGLE
COMMON b AS SINGLE
COMMON c AS DOUBLE
INPUT a, b
c = a / b
PRINT a; b
PRINT c
END
```

【結 果】

```
? 2354.25, 12.85436
2354.25 12.85436
183.1479797363281
```

14 章 算術関数

SIN, COS, TAN, ATN, SQR, ABS, LOG, EXP, CINT, CLNG,
INT, FIX, SGN, RANDOMIZE, RND, TIMER

【例題 97】 3 角関数

角 30 度の sin, cos, tan の値を求めるプログラムをつくれ。

【解 説】 ◎ 3 角関数の sin, cos, tan を求めます。

① 3 角関数の値は次の形で求まります。ただし、 x はラジアン単位です。

正弦 $\sin(x)$ $\sin(x/180*3.14159)$ で x 度の場合が求まる

余弦 $\cos(x)$ $\cos(x/180*3.14159)$ で x 度の場合が求まる

正接 $\tan(x)$ $\tan(x/180*3.14159)$ で x 度の場合が求まる

【プログラム例】

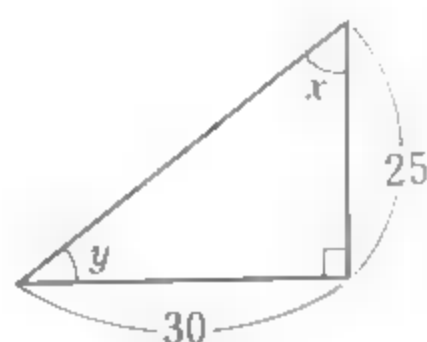
```
x = 30 / 180 * 3.14159 ——— 30 度のラジアンの値を x に代入
PRINT SIN(x) ————— SIN, COS, TAN の値を表示
PRINT COS(x)
PRINT TAN(x)
END
```

【結 果】

```
.49999996
.8660256
.5773497
```

【例題 98】 アークタンジェント

下図の角 x 度を求めるプログラムをつくれ。



【解 説】 ◎ アークタンジェントを求めます。

① アークタンジェントは次の式で求めます。ただし、結果はラジアンです。

ATAN(x)

【プログラム例】

```

xx = ATN(25 / 30) —— アークタンジェントの値を求める
x = xx * 180 / 3.14159 —— xx ラジアンを x 度に変換する
PRINT x; "°"
END

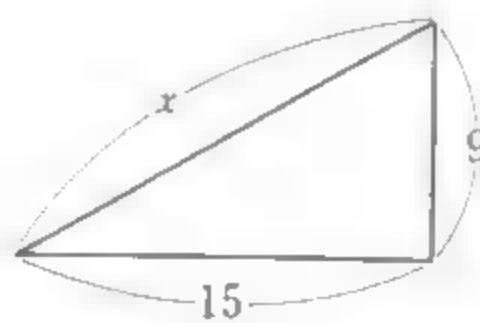
```

【結 果】

39.8056 °

【例題 99】 平方根

下図の x の長さを求めるプログラムをつくれ。



【解 説】 ◎ 平方根を求めます。

- ① 平方根は次の形で求めます。

$\text{SQR}(x)$

- ② したがって、3 角形の斜辺の長さは、 $\sqrt{15^2+9^2}$ ですから $\text{SQR}(15^2+9^2)$ で求められます。

【プログラム例】

```

x = SQR(15 * 15 + 9 * 9) ——  $\sqrt{15^2+9^2}$  を求める
PRINT x
END

```

【結 果】

17.49286

【例題 100】 絶対値

整数を 2 つ入力して、その差を求めるプログラムをつくれ。

【解 説】 ◎ 絶対値を求めます。

- ① 2 数の差は、2 数の差の絶対値です。絶対値は次の式で求めます。

$\text{ABS}(x)$

- ② したがって、 a と b の差は次の式で求められます。

$\text{ABS}(a-b)$ $\text{ABS}(b-a)$ でも同じです。

【プログラム例】

```

INPUT a, b —— a と b の入力
PRINT ABS(a - b) —— a と b の差を求める
END

```

【結 果】

? 5, 2	5 と 2 を入力の場合
3	
? 2, 5	2 と 5 を入力の場合
3	

【例題 101】 対 数

100 万円を年利率 4.3%で預金したとき 200 万円になるのに何年かかるかを求めるプログラムをつくれ。

【解 説】 ◎ 対数を求めます。

- ① 100 万円を 4.3%の年利率で預金したとき y 年で 200 万円になるとします。すると次の式となります。

$$100 * (1 + 0.043)^y = 200 \quad \cdots (1)$$

したがって、 $(1 + 0.043)^y = 2 \quad \cdots (2)$

対数をとると、 $y \log(1.043) = \log 2$

したがって、 $y = \log 2 / \log(1.043)$

- ② 対数は $\log(x)$ で表わされます。

したがって、 $y \log(1.043) = \log(2)$ となり、

$$y = \frac{\log(2)}{\log(1.043)}$$

で y が求まります。

なお、 y 年は整数ですから切りあげで y を求めます。

- ③ なお、常用対数と自然対数は次の関係で求められます。

$$\log_{10} x = \frac{\log_e x}{\log_e 10}$$

$\text{LOG}(x)$ は e を底とする自然対数の値です。

- ④ $\text{LOG}(x)$ の x は正の値で、 x が単精度実数のときは結果も単精度実数、 x が倍精度実数のとき結果も倍精度実数となります。

- ⑤ e の値はおよそ 2.718282 です。

【プログラム例】

```

y = LOG(2) / LOG(1.043)——— y 年を求める
PRINT INT(y + .99999)——— y 年を切りあげる
END

```

【結 果】

17

【例題 102】 指数関数

$e^{3.87}$ を求めるプログラムをつくれ.

【解 説】 ◎ 指数関数を求めます.

- ① $e^{3.87}$ は EXP(3.87) で求められます.

【プログラム例】

```
y = EXP(3.87) —— 指数関数値を求める
PRINT y
END
```

【結 果】

47.94238

【例題 103】 四捨五入、切りあげ、切りすて

15.38 と 358.29645 を各々小数点第 2 位と第 4 位で四捨五入、切りあげ、切りすてで各々小数第 1 位と第 3 位まで求めるプログラムをつくれ.

【解 説】 ◎ 四捨五入、切りあげ、切りすてをします.

- ① 四捨五入は CINT, CLNG を使います. これらは小数点以下を四捨五入して, それぞれ整数, 長整数とします.
- ② 任意の桁で四捨五入するには次のようにします. CINT を例とします. 10^n を掛けたり, 割ったりした後 CINT を行い, その後 10^n を掛けたり, 割ったりして元へ戻します.

```
15.38      CINT(15.38) → 15
            CINT(15.38*10)/10 → 15.4
            CINT(15.38÷10)*10 → 20
```

- ③ 切りすては int を使います. int は切りすてで整数化します. 任意の桁位置での切りすては四捨五入と同じ考えです.

```
15.38      INT(15.38) → 15
            INT(15.38*10)/10 → 15.3
            INT(15.38÷10)*10 → 10
```

- ④ 切りあげは int を応用します. 切りあげを行う桁が 1 以上のときに切りあげますから, その桁に 9 を加えてから切りすてすれば同じ結果が得られます.

```
15.38      INT(15.38+0.9) → 16
            INT(15.38*10+0.9)/10 → 15.4
            INT(15.38÷10+0.9)*10 → 20
```

【プログラム例】

```

x = 15.38
y# = 358.29645#
x1 = CINT(x * 10) / 10 —— 小数第 2 位で四捨五入
x2 = INT(x * 10 + .9) / 10 —— 小数第 2 位で切りあげ
x3 = INT(x * 10) / 10 —— 小数第 2 位で切りすて
y1 = CLNG(y# * 1000) / 1000 —— 小数第 4 位で四捨五入
y2 = INT(y# * 1000 + .9) / 1000 —— 小数第 4 位で切りあげ
y3 = INT(y# * 1000) / 1000 —— 小数第 4 位で切りすて
PRINT x1, x2, x3 —— 結果の表示
PRINT y1, y2, y3
END

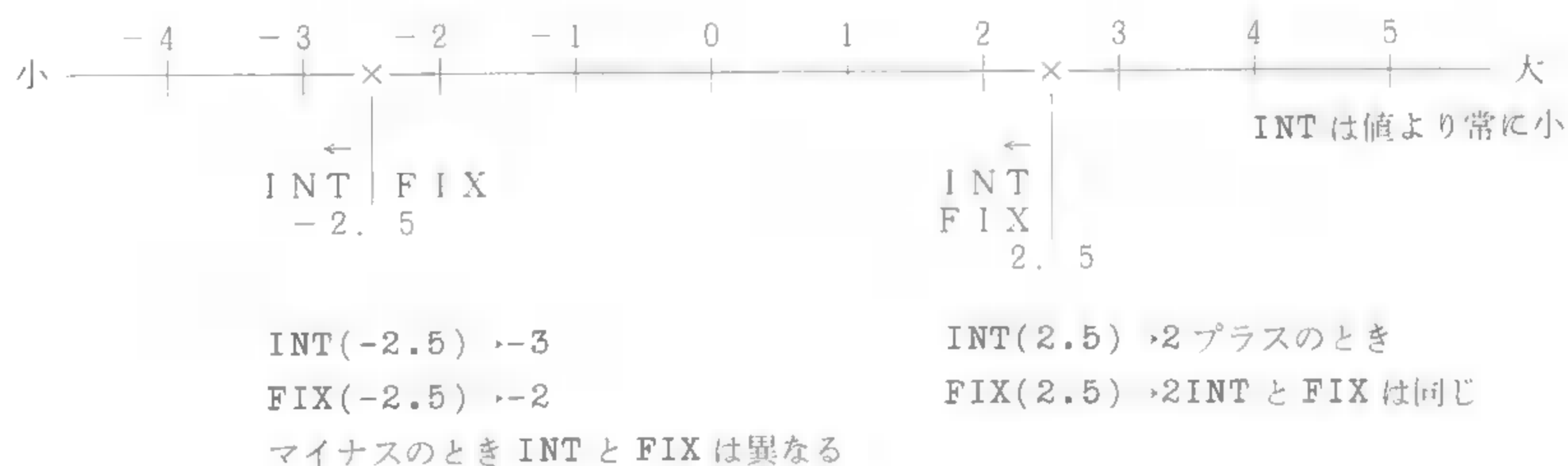
```

【結 果】

15.4	15.4	15.3
358.296	358.297	358.296

【例題 104】 整数部分の値のとり出し

-35.86 の FIX 値と 15.38 の INT 値をとり出すプログラムをつくれ。



【解 説】 ◎ 整数部の値をとり出します。

- ① FIX は INT と整数部をとり出す点で似ていますが、マイナスの場合では異なります。INT はその値をこえない最大の整数値です。FIX は整数部の値です。

【プログラム例】

```

x = -35.86
y = 15.38
x1 = FIX(x) —— -35.86 の整数部の値
y1 = INT(y) —— 15.38 をこえない最大の整数値
PRINT x1, y1
END

```

【結 果】

-35	15
-----	----

【例題 105】 SGN

$y=ax^2+3x+c$ の式で a の値を入力し、プラスのとき"シタ ガ トツ", $a<0$ のとき"ウエ ガ トツ", $a=0$ のとき"チョクセン"と表示するプログラムをつくれ.

【解 説】 ◎ SGN は値の範囲によって 1, 0, -1 を与えます.

- ① 2 次式の値は a の値がプラス、マイナス、0 によって次図のようになります.



$a > 0$



$a < 0$



$a = 0$

- ② a を入力し、 $\text{SGN}(a)$ によって、 a がプラスのとき $\text{SGN}(a)$ は 1, マイナスのとき -1, 0 のとき 0 となります. したがって、 a の値によって SGN の値を 3 つに分けて分岐します.

【プログラム例】

```

INPUT a ----- a の入力
s = SGN(a) ----- a の SGN を s に求める
IF s > 0 THEN ----- s の値によって表示を選ぶ
    PRINT "シタ ガ トツ"
ELSE
    IF s = 0 THEN
        PRINT "チョクセン"
    ELSE
        PRINT "ウエ ガ トツ"
    END IF
END IF
END IF
END

```

【結 果】

? 5	5 を入力の例
シタ ガ トツ	
? -4	-4 を入力の例
ウエ ガ トツ	
? 0	0 を入力の例
チョクセン	

【例題 106】 乱 数

1000~9999 の乱数を 100 個つくり、1000 台、2000 台、...9000 台が各々いくつできたかを求めるプログラムをつくれ.

【解 説】 ◎ 乱数をつくります.

- ① 乱数は RND でつくりませんが、これだけでは同じ乱数系列を使うため、プログラムを実行するごとに同じ値がくり返しできます. これは、シミュレーションなどで、システムの構成を変えて再びトライする場合には有効ですが、ゲームでは同じことのくり返しでおもしろくありません. 乱数系列を変える命令が RANDOMIZE です.

② RANDOMIZE は実行されると数値を聞いてきて、値を入力するとその値の乱数系列をつくり、つくれる乱数は新しい系列となります。

③ 自動的に新しい乱数系列をつくりたいときは TIMER を利用して、刻々かわる時間の数値を利用してつくり出します。

④ RND(x) は x の値によって 0 から 1 未満の乱数をつくり、x が正または RND のみのときは次の乱数をつくり、0 のときは 1 つ前の乱数、負のときは同じ乱数をつくり。

RND(5) の値を a とする

RND(3) の値を b とする

RND(4) の値を c とする

RND(0) は b となる

RND(5) の値を d とする

RND(-3) は d となる

⑤ 次に乱数は 0 から 1 未満の値ですからこれを使って 1000 から 9999 の数にします。この方法は次のとおりです。

$\text{INT}(\text{RND}(5) * 9000) + 1000$

0 から 8999 の値となる

一般化すると a から b の間の乱数をつくる時は次のようになります。

$\text{INT}(\text{RND}(5) * (b - a + 1)) + a$

【プログラム例】

```

DIM z%(1 TO 9)
RANDOMIZE (TIMER)
FOR i = 1 TO 100
    x% = INT(RND(5) * 9000) + 1000
    y% = INT(x% / 1000)
    z%(y%) = z%(y%) + 1
NEXT i
FOR i = 1 TO 9
    PRINT i * 1000; "ダ イ="; z%(i)
NEXT i
END

```

100 個の 1000~9999 の乱数をつくりカウントする

結果の表示

【結 果】

```

1000 ダ イ= 13
2000 ダ イ= 12
3000 ダ イ= 9
4000 ダ イ= 8
5000 ダ イ= 13
6000 ダ イ= 7
7000 ダ イ= 13
8000 ダ イ= 10
9000 ダ イ= 15

```

例①

```

1000 ダ イ= 10
2000 ダ イ= 11
3000 ダ イ= 9
4000 ダ イ= 11
5000 ダ イ= 14

```

例②

6000 タイ= 9
 7000 タイ= 11
 8000 タイ= 15
 9000 タイ= 10

【例題 107】 乱数を使ったシミュレーション

3 割 2 分 5 厘の成績の打者の今後 30 打席のヒットの確率を求めるプログラムをつくれ。

【解 説】 ◎ 乱数を使ったシミュレーション

- ① 乱数を使ったシミュレーションを行う例を 1 つ示します。
- ② 3 割 2 分 5 厘の打率の打者のヒットの確率は .325 です。したがって、乱数が 0～1 未満とすれば乱数をつかって 0～.325 のときはヒット、それ以外ではアウトとすればよいことになります。
- ③ 乱数を 30 回つakって、0～.325 の数を数え、30 で割れば、今後 30 打席の打率となります。

【プログラム例】

```

RANDOMIZE (TIMER)
FOR i = 1 TO 30
    x = RND(5)
    IF x <= .325 THEN d = d + 1
NEXT i
PRINT d / 30
END
    
```

乱数をつくり、325 以下の
とき数を数える

打率の計算

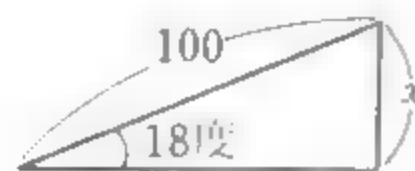
【結 果】

.2666667
 .4333333
 .2
 .3
 .2333333

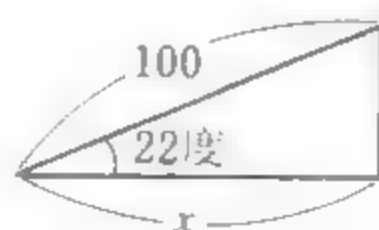
〔例〕 5 回実行したもの

〔演習問題〕

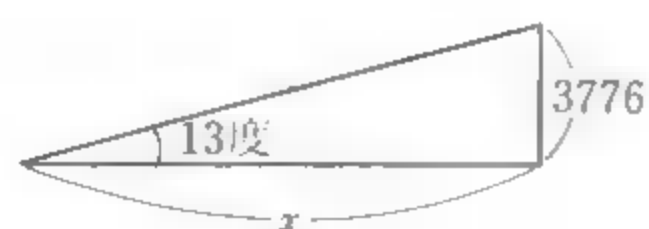
- (113) 下図の x を求めるプログラムをつくれ。



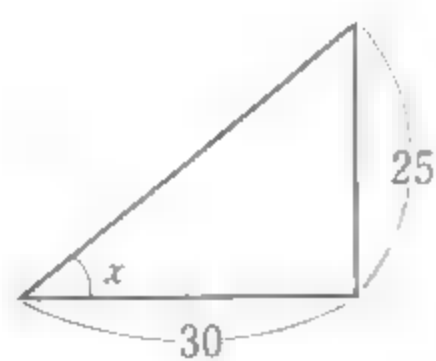
- (114) 下図の x を求めるプログラムをつくれ。



- (115) 下図の x を求めるプログラムをつくれ。



- (116) 下図の x 度と y 度を求めるプログラムをつくれ.



- (117) $3x^2 + 2x - 3 = 0$ の式での x の値を求めるプログラムをつくれ.
- (118) 小数点以下 3 桁の数を 1 つ入力し、小数第 3 位を四捨五入、切りあげ、切りすてで小数点以下第 2 位までの値を求めるプログラムをつくれ.
- (119) 150 万円を年利率 5.1% の複利預金をしたとき、元利合計が 3 倍となるのに必要な年数を求めるプログラムをつくれ.
- (120) 基準地からの距離を 2 つ入力して、その地区間の距離を求めるプログラムをつくれ.
- (121) サイコロを 10 回ふって 1~6 の目の出かたをシミュレートするプログラムをつくれ.

〔解 答〕

(113) 【プログラム例】

```
x = 18 / 180 * 3.14159 —— 18 度のラジアン値
PRINT 100 * SIN(x) —— x ラジアン の SIN 値 * 100
END
```

【結 果】

30.90168

(114) 【プログラム例】

```
x = 22 / 180 * 3.14159 —— 22 度のラジアン値
PRINT 100 * COS(x) —— x ラジアン の COS 値 * 100
END
```

【結 果】

92.7184

(115) 【プログラム例】

```
x = 13 / 180 * 3.14159 —— 13 度のラジアン値
PRINT 3776 / TAN(x) —— x ラジアン の TAN 値 * 3776
END
```

【結 果】

16355.67

(116) 【プログラム例】

```

x = ATN(30 / 25) ----- アークタンジェント
y = ATN(25 / 30) -----
x1 = x * 180 / 3.14159 ----- x, y ラジアンを度に変換
y1 = y * 180 / 3.14159 -----
PRINT x1; "°"
PRINT y1; "°"
END

```

【結 果】

50.19447 °
39.8056 °

(117) 【プログラム例】

```

z = SQR(2 * 2 - 4 * 3 * (-3)) -----  $\sqrt{b^2 - 4ac}$ 
x1 = (-2 + z) / (2 * 3) -----  $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$ 
x2 = (-2 - z) / (2 * 3) -----  $\frac{-b - \sqrt{b^2 - 4ac}}{2a}$ 
PRINT x1, x2
END

```

【結 果】

.7207592 -1.387426

(118) 【プログラム例】

```

INPUT x ----- x を入力
x1 = CINT(x * 100) / 100 ----- 小数第 3 位で四捨五入
x2 = INT(x * 100 + .9) / 100 ----- 小数第 3 位で切りあげ
x3 = INT(x * 100) / 100 ----- 小数第 3 位で切りすて
PRINT x1, x2, x3
END

```

【結 果】

? 1.354			1.354 を入力の例
1.35	1.36	1.35	
? 12.459			12.459 を入力の例
12.46	12.46	12.45	

(119) 【プログラム例】

```

y = LOG(3) / LOG(1.051) ----- y 年を求める
PRINT INT(y + .99999) ----- y 年を切りあげる
END

```

【結 果】

23

(120) 【プログラム例】

```

INPUT a, b
PRINT "a="; a
PRINT "b="; b
PRINT "キヨリ="; ABS(a - b)
END

```

a, b の入力
a と b の差を求める

【結 果】

```

? 152,987          152 と 987 を入力の例
a= 152
b= 987
キヨリ= 835
? 562,157          562 と 157 を入力の例
a= 562
b= 157
キヨリ= 405

```

(121) 【プログラム例】

```

RANDOMIZE (TIMER)
y = 1 / 6
FOR i = 1 TO 10
  x = RND(5)
  IF x <= y THEN PRINT 1;
  IF x > y AND x <= 2 * y THEN PRINT 2;
  IF x > 2 * y AND x <= 3 * y THEN PRINT 3;
  IF x > 3 * y AND x <= 4 * y THEN PRINT 4;
  IF x > 4 * y AND x <= 5 * y THEN PRINT 5;
  IF x > 5 * y THEN PRINT 6;
NEXT i
PRINT
END

```

1～6 を乱数でつくり各々を表示する

【結 果】

```

5 6 1 3 6 3 3 5 5 6
5 1 4 5 1 6 2 2 2 4
4 3 5 2 4 4 6 2 2 6

```

3 回実行した例

15 章 論理演算

AND, OR, XOR, EQV, IMP, NOT

【例題 108】 AND

15 と 3 の AND, -15 と 3 の AND を求めるプログラムをつくれ.

【解 説】 ◎ AND を求めます.

① 論理演算はビット操作をするときにも使います. 16 ビットで先頭を符号とし, プラスのとき 0, マイナスのとき 1 とします.

② 15, 3 を 16 ビット, 先頭を符号で表わすと

	符号	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
A=15	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
B=3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります. したがって,

A AND B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
---------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

となります. したがって, 15 AND 3 は, 3 です.

③ -15, 3 を 16 ビット, 先頭を符号で表わすと

	符号	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
A=-15	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	1
B=3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります. したがって,

A AND B	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
---------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

となります. したがって, -15 AND 3 は, 1 です.

マイナスの値は補数で表わされます.

【プログラム例】

```
x1% = 15
y1% = 3
x2% = -15
PRINT x1% AND y1%———15 と 3 の AND
PRINT x2% AND y1%———-15 と 3 の AND
END
```

【結 果】

3
1

【例題 109】 OR

15 と 3 の OR と -15, 3 の OR を求めるプログラムをつくれ.

【解 説】 ◎ OR を求めます.

① 15 および 3 を 16 ビットで, 先頭を符号で表わすと

	符号	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
A=15	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
B=3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります. したがって,

A OR B	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
--------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

となります. したがって, 15 OR 3 は, 15 です.

② -15 と 3 を 16 ビット, 先頭を符号で表わすと

	符号	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
A=-15	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
B=3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります. したがって,

A OR B	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1
--------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

です. これは -13 です.

【プログラム例】

```

x1% = 15
y1% = 3
x2% = -15
PRINT x1% OR y1% —— 15 と 3 の OR
PRINT x2% OR y1% —— -15 と 3 の OR
END

```

【結 果】

```

15
-13

```

【例題 110】 XOR

15 と 3 の XOR, -15 と 3 の XOR を求めるプログラムをつくれ.

【解 説】 ◎ XOR を求めます.

① 15, 3 を 16 ビット, 先頭を符号で表わすと

	符号	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
A=15	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
B=3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります。したがって、

A XOR B 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0

となります。したがって、15 XOR 3 は、12 です。

- ② -15 と 3 を 16 ビット、先頭を符号で表わすと

	符号	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
A=-15	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
B=3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります。したがって、

A XOR B 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

です。これは-14 です。

【プログラム例】

```

x1% = 15
y1% = 3
x2% = -15
PRINT x1% XOR y1%———15 と 3 の XOR
PRINT x2% XOR y1%———-15 と 3 の XOR
END

```

【結 果】

12
-14

【例題 111】 EQV

15 と 3 の EQV、-15 と 3 の EQV を求めるプログラムをつくれ。

【解 説】 ◎ EQV を求めます。

- ① 15 と 3 を 16 ビット、先頭を符号で表わすと

	符号	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
A=15	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
B=3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります。したがって、

A EQV B 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1

です。これは-13 です。

- ② -15, 3 を 16 ビット、先頭を符号で表わすと

	符号	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰
A=-15	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
B=3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります。したがって、

A EQV B 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1

となります。したがって、-15 EQV 3 は、13 です。

【プログラム例】

```

x1% = 15
y1% = 3
x2% = -15
PRINT x1% EQV y1% —— 15 と 3 の EQV
PRINT x2% EQV y1% —— -15 と 3 の EQV
END

```

【結 果】

```

-13
13

```

【例題 112】 IMP

15 と 3 の IMP, -15 と 3 の IMP を求めるプログラムをつくれ.

【解 説】 ◎ IMP を求めます.

(1) 15, 3 を 16 ビット, 先頭を符号で表わすと

	符号	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
A=15	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
B=3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります. したがって,

A IMP B	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1
---------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

です. これは -13 です.

(2) -15 と 3 を 16 ビット, 先頭を符号で表わすと

	符号	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
A=-15 は,	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
B=3 は,	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

となります. したがって,

A IMP B は,	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
------------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

となります. すなわち, -15 IMP 3 は, 15 です.

【プログラム例】

```

x1% = 15
y1% = 3
x2% = -15
PRINT x1% IMP y1% —— 15 と 3 の IMP
PRINT x2% IMP y1% —— -15 と 3 の IMP
END

```

【結 果】

```

-13
15

```

【例題 113】 NOT

15 の NOT と -15 の NOT を求めるプログラムをつくれ.

【解 説】 ◎ NOT を求めます.

① 15 を 16 ビットで、先頭を符号で表わすと

	符号	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
A=15	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

です. したがって,

NOT A	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
-------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

です. これは -16 です.

② -15 を 16 ビットで、先頭を符号で表わすと

	符号	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
A=-15	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1

です. したがって,

NOT A	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0
-------	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

です. したがって, 14 です.

【プログラム例】

```

x1% = 15
x2% = -15
PRINT NOT x1% —— 15 の NOT
PRINT NOT x2% —— -15 の NOT
END

```

【結 果】

```

-16
14

```

〔演習問題〕

- (122) 189 と 25 の AND を求めるプログラムをつくれ.
- (123) 29 と -8 の各々の NOT の OR を求めるプログラムをつくれ.
- (124) 80 と 73 の IMP と 80 と 73 の EQV の OR を求めるプログラムをつくれ.
- (125) -15 と -85 の XOR を求めるプログラムをつくれ.

〔解 答〕

(122) 【プログラム例】

```
x1% = 189
x2% = 25
PRINT x1% AND x2%
END
```

【結 果】

25

(123) 【プログラム例】

```
x1% = 29
x2% = -8
PRINT (NOT (x1%)) OR (NOT (x2%))——(29 の NOT) と (-8 の NOT) の OR
END
```

【結 果】

-25

(124) 【プログラム例】

```
x1% = 80
x2% = 73
PRINT (x1% IMP x2%) OR (x1% EQV x2%)——(80 IMP 73) と (80 EQV 73) の OR
END
```

【結 果】

-17

(125) 【プログラム例】

```
x1% = -15
x2% = -85
PRINT x1% XOR x2%——-15 と -85 の XOR
END
```

【結 果】

90

16 章 文字列関数

LEFT\$, RIGHT\$, MID\$, LEN, SPACE\$, LTRIM\$, RTRIM\$,
MID\$=, INSTR, LCASE\$, UCASE\$, STRING\$

【例題 114】 文字列のとり出し

"Microsoft Quick Basic"の中から"Microsoft", "Basic", "Quick"をとり出して表示するプログラムをつくれ.

【解 説】 ◎ 文字列の中から任意の文字列をとり出します.

- ① a\$を"Microsoft Quick Basic"とします.
- ② LEFT\$, RIGHT\$, MID\$は各々a\$の中から文字列をとり出します.

Microsoft
└────────┘

Quick
└───┘

Basic
└───┘

LEFT\$(a\$, 9)

MID\$(a\$, 11, 5)

RIGHT\$(a\$, 5)

左側 9 文字

11 番目から 5 文字

右側 5 文字

【プログラム例】

```
a$ = "Microsoft Quick Basic"—— 文字列を a$に代入
b$ = LEFT$(a$, 9)—— 左側 9 文字を b$に代入
c$ = RIGHT$(a$, 5)—— 右側 5 文字を c$に代入
d$ = MID$(a$, 11, 5)—— 11 番目から 5 文字を d$に代入
PRINT b$—— b$, c$, d$の表示
PRINT c$
PRINT d$
END
```

【結 果】

```
Microsoft
Basic
Quick
```

【例題 115】 文字列の長さ

"Microsoft Quick Basic"を 1 文字ずつとり出して表示するプログラムをつくれ.

【解 説】 ◎ 文字列の長さを求めます.

- ① a\$に"Microsoft Quick Basic"を代入します.
- ② LEN(a\$)は文字列 a\$の長さ, 文字数を求めます.

【プログラム例】

```

a$ = "Microsoft Quick Basic" —— 文字列を a$ に代入
b = LEN(a$) —— 文字数を b に代入
FOR i = 1 TO b —— 1 番目から b 番目まで 1 文字ずつとり
    c$ = MID$(a$, i, 1) —— 出して c$ へ代入し、c$ を順に表示
    PRINT c$;
NEXT i —— 改行
PRINT
END

```

【結 果】

Microsoft Quick Basic

【例題 116】 空 白

5 桁 3 行目から 5 つの文字列を入力して、6, 8, 10, 12, 14 行の 5 桁目からそれぞれ表示するプログラムをつくれ。

【解 説】 ◎ 空白を表示します。

- ① 文字列を入力する前に、同じ行列から空白を表示して、入力する位置を空白にします。前回入力した文字列が消え、同じ行列から文字列の入力ができます。
- ② 空白は `space$(x)` で `x` 個の空白を持つ文字列をつくります。

【プログラム例】

```

DIM a$(5) —— 5 文字列の入る配列変数の宣言
FOR i = 1 TO 5 —— 5 文字列の入力
    LOCATE 3, 5: PRINT SPACE$(30) —— 5 桁 3 行目から 30 空白文字をかく
    LOCATE 3, 5: INPUT a$(i) —— 5 桁 3 行目から文字列の入力
NEXT i
FOR i = 1 TO 5 —— 表示
    LOCATE 4 + i * 2, 5: PRINT a$(i)
NEXT i
END

```

【結 果】

? C 左の 5 文字列を入力した例

Microsoft

Quick

Basic

Quick

C

【例題 117】 空白の除去

a\$を"Microsoft ", b\$を"Quick", c\$を"Basic" (□は空白) として
a\$+b\$+c\$と各文字列の左側の空白をとってa\$+b\$+c\$を表示するプログラムをつくれ。

【解説】 ◎ 空白を除去します。

- ① LTRIM\$(a\$)で文字列 a\$の左側 (先頭) の空白をとります。

```
a$                                LTRIM$(a$)
"[Microsoft]"                    "Microsoft"
```

- ② RTRIM\$(a\$)は文字列の右側の空白をとります。これは〔演習問題〕(128)で扱います。

【プログラム例】

```
a$ = " Microsoft "
b$ = " Quick "
c$ = " Basic "
PRINT a$ + b$ + c$
PRINT LTRIM$(a$) + LTRIM$(b$) + LTRIM$(c$)
END
```

【結果】

Microsoft Quick Basic
Microsoft Quick Basic

【例題 118】 文字列の入れかえ

a\$を"Microsoft Quick Basic"とし、次に"Basic"を"C□□□□" (□は空白)で
入れかえて表示するプログラムをつくれ。

【解 説】 ◎ 文字列の 1 部の入れかえをします。

- ① a\$を"Microsoft Quick Basic"とします。 b\$を"C[][][]" ([]は空白) とします。
- ② "Basic"の部分を"C[][][]"と入れかえます。

Microsoft Quick Basic

55555

MID\$(a\$,17,5)=b\$

a\$の17番目から5文字をb\$で入れかえ

【プログラム例】

```
a$ = "Microsoft Quick Basic"——— a$に文字列を代入
b$ = "C"——— b$に文字列を代入
MID$(a$, 17, 5) = b$——— a$の 17 番目から 5 文字を b$で入れかえ
PRINT a$——— a$の表示
END
```

【結 果】

Microsoft Quick C

【例題 119】 検 索

a\$を"Microsoft Quick Basic", b\$を"Microsoft", c\$を"soft", d\$を"Quick"として, a\$の最初から b\$, c\$, d\$があるかどうか, また a\$の 17 番目以降に d\$があるかどうかを検索するプログラムをつくれ.

【解 説】 ◎ 文字列の検索をします.

- ① a\$を"Microsoft Quick Basic", b\$を"Microsoft", c\$を"soft", d\$を"Quick"とします.
- ② INSTR(1,a\$,b\$)はa\$の中にb\$があるかどうかを検索します. このとき1番目からさがします. 見つかった場合, それが何番目からあるか, その番号を与えます. "Microsoft Quick Basic"の中に"Microsoft"は1番目からありますのでINSTR(1,a\$,b\$)は1です. 見つからないときは0です.

【プログラム例】

```

a$ = "Microsoft Quick Basic"
b$ = "Microsoft"
c$ = "soft"
d$ = "Quick"
PRINT INSTR(1, a$, b$)
PRINT INSTR(1, a$, c$)
PRINT INSTR(1, a$, d$)
PRINT INSTR(17, a$, d$)
END

```

文字列を a\$~d\$ に代入
a\$の中で b\$, c\$, d\$ を最初から検索,
a\$の中で d\$ を 17 番目から検索

【結 果】

```

1
6
11
0

```

【例題 120】 小文字化

"Microsoft Quick Basic"を小文字で表示するプログラムをつくれ.

【解 説】 ◎ 大文字を小文字化します.

- ① LCASE\$(a\$)はa\$の中の大文字を小文字にします.
- ② UCASE\$(a\$)はa\$の中の小文字を大文字にします. これは〔演習問題〕(131)で扱います.

【プログラム例】

```

a$ = "Microsoft Quick Basic"
PRINT LCASE$(a$)
END

```

文字列を a\$ に代入
a\$の大文字を小文字にして表示

【結 果】

microsoft quick basic

【例題 121】 特定の文字の列

"A"を10個つないだ文字列をつくるプログラムをつくれ.

【解 説】 ◎ 特定の文字を指定した数つないだ文字列をつくります.

- ① STRING\$(a,b)ではアスキーコードをbとする文字・記号をa個つないだ文字列をつくります.

STRING\$(10,65)

65すなわち"A"を10個つないだ文字列

STRING\$(10,"A")でもよい.

STRING\$(10,&H41)でもよい.

- ② bが文字列の場合, 先頭1文字となります.

【プログラム例】

```
a$ = STRING$(10, 65) —— "A"を10個つないだ文字列をa$に代入
PRINT a$ —— a$の表示
END
```

【結 果】

AAAAAAAAAA

【例題 122】 グラフ

次の表のデータをグラフとするプログラムをつくれ.

成績

名前	点
川上	80
山下	90
足立	56
瀬島	74

【解 説】 ◎ STRING\$を使ってグラフをつくります.

- ① STRING\$を使って簡単なグラフをつくります.
- ② 100点を"*"50個で表わすとすれば各人の点数/2個の*をつないだ文字列をつくって表示すれば各人の点を横棒グラフで表わせます.

例

100点 STRING\$(100/2,"*") *が50個の文字列

90 点 `STRING$(90/2, "*")` *が 45 個の文字列

x 点 `STRING$(x/2, "*")` *が x/2 個の文字列

③ 目盛りと軸は "+" と "-" を 4 つつないだ文字列を 10 個と最後に "+" をつけます。

`b$ = "+" + STRING$(4, "-")` とすると `b$` が "+----" となります。

+----+----+----+-----+----+
 0 b\$ 10 b\$ 20 b\$ 90 b\$ 100点を示す

【プログラム例】

```

DIM a$(4), s$(4)
FOR i = 1 TO 4
    READ a$(i), x
    s$(i) = STRING$(x / 2, "*")
NEXT i
b$ = "+" + STRING$(4, "-")
LOCATE 3, 20: PRINT "4ニ / セイセキ(テン)"
FOR i = 1 TO 4
    LOCATE 3 + 2 * i, 10: PRINT a$(i)
    LOCATE 3 + 2 * i, 16: PRINT "+"
    LOCATE 3 + 2 * i, 17: PRINT s$(i)
NEXT i
FOR i = 1 TO 10
    LOCATE 13, 16 + (i - 1) * 5: PRINT b$
NEXT i
LOCATE 13, 16 + (i - 1) * 5: PRINT "+"
FOR i = 0 TO 100 STEP 10
    LOCATE 14, 15 + i / 10 * 5: PRINT i
NEXT i
END
DATA カワカミ, 80, ヤマシタ, 90, アタチ, 56, セシマ, 74
  
```

`a$` に名前, `s$` に点数にあらう文字列が入る
 名前と点数を読み, 名前を `a$`, 点数にあらう
 "*" の文字列を `s$` に代入
`b$` に "+----" をつくって代入
 標題
 名前, +, 点数にあらう*を表示
 軸の表示
 点数を表示

【結 果】

```

      4ニ / セイセキ(テン)

カワカミ  ++++++
ヤマシタ  ++++++
アタチ    ++++++
セシマ    ++++++

+----+----+----+-----+----+
0      10     20     30     40     50     60     70     80     90    100
  
```

〔演習問題〕

- (126) "マイクロソフト クイック ベーシック"の中から, "マイクロソフト", "ベーシック", "クイック"を各々とり出して表示するプログラムをつくれ。
- (127) 文字列を 2 つ `a$`, `b$` に代入し, 長い方を表示するプログラムをつくれ。

- (128) a\$を"Microsoft", b\$を"Quick", c\$を"Basic"として a\$+b\$+c\$ と、各々の文字列の右側の空白をとって a\$+b\$+c\$を表示するプログラムをつくれ。
- (129) a\$を"ダイレクトモード"とし、b\$を"ジッコウ"とし a\$の8番目から5文字を b\$で 入れかえるプログラムをつくれ。
- (130) a\$を"Microsoft Quick Basic"とし、文字列を b\$に代入して a\$の中に b\$があれば a\$を、なければ b\$を表示するプログラムをつくれ。
- (131) "Microsoft Quick Basic"を大文字にして表示するプログラムをつくれ。
- (132) #を10個つないだ文字列を表示するプログラムをつくれ。
- (133) "Microsoft"の先頭1文字Mを20個つないだ文字列を表示するプログラムをつくれ。
- (134) 下のデータを棒グラフにするプログラムをつくれ。

5社の初任給

社 名	初任給 (万円)
中西 S S	15
日本鉄工	19
東京 S S	22
島 商 事	12
本多印刷	10

- (135) "Microsoft"の各文字の後に2スペースをつけて表示するプログラムをつくれ。
- (136) "USA"を画面左から右へ動かすプログラムをつくれ。

〔 解 答 〕

(126) 【プログラム例】

```

a$ = "マイクロソフト クイック ベーシック" —— 文字列を a$ に代入
b$ = LEFT$(a$, 7) —— 左側 7 文字を b$ に代入
c$ = RIGHT$(a$, 6) —— 右側 6 文字を c$ に代入
d$ = MID$(a$, 9, 4) —— 9 番目から 4 文字を d$ に代入
PRINT b$ —— b$, c$, d$ の表示
PRINT c$
PRINT d$
END

```

【結 果】

```

マイクロソフト
ベーシック
クイック

```

(127) 【プログラム例】

```

INPUT a$ —— 文字列を入力して a$ へ代入
INPUT b$ —— 文字列を入力して b$ へ代入

```

```

11 = LEN(a$) —— a$の文字数を 11 に代入
12 = LEN(b$) —— b$の文字数を 12 に代入
IF 11 >= 12 THEN —— 長い方の文字列を表示
    PRINT a$
ELSE
    PRINT b$
END IF
END

```

【結 果】

```

? length          length と width を入力の場合
? width
length
? dog              dog と window を入力の場合
? window
window

```

(128) 【プログラム例】

```

a$ = " Microsoft " —— a$, b$, c$に各文字列を代入
b$ = " Quick "
c$ = " Basic "
PRINT a$ + b$ + c$ —— a$+b$+c$を表示
PRINT RTRIM$(a$) + RTRIM$(b$) + RTRIM$(c$) —— 右側の空白を除去して a$+b$+c$を表示
END

```

【結 果】

```

Microsoft Quick Basic
Microsoft Quick Basic

```

(129) 【プログラム例】

```

a$ = "ダイレクト モード" —— a$に文字列を代入
b$ = "ジッコウ" —— b$に文字列を代入
MID$(a$, 8, 5) = b$ —— a$の 8 番目から 5 文字を b$で入れかえ
PRINT a$ —— a$の表示
END

```

【結 果】

```

ダイレクト シェッコウ

```

(130) 【プログラム例】

```

a$ = "Microsoft Quick Basic" —— 文字列を a$に代入
INPUT b$ —— 文字列を b$に代入
IF INSTR(1, a$, b$) <> 0 THEN —— a$の中に b$があれば a$を表示, なければ b$を表示
    PRINT a$
ELSE
    PRINT b$
END IF
END

```

【結 果】

```
? Basic                      Basic を入力の場合
Microsoft Quick Basic
? basic                      basic を入力の場合
basic
```

(131) 【プログラム例】

```
a$ = "Microsoft Quick Basic" —— 文字列を a$ に代入
PRINT UCASE$(a$) —— a$ を大文字にして表示
END
```

【結 果】

```
MICROSOFT QUICK BASIC
```

(132) 【プログラム例】

```
a$ = STRING$(10, "#") —— "#" を 10 個つないだ文字列を a$ に代入
PRINT a$ —— a$ の表示
END
```

【結 果】

```
#####
```

(133) 【プログラム例】

```
a$ = STRING$(20, "M") —— "M" を 20 個つないだ文字列を a$ に代入
PRINT a$ —— a$ の表示
END
```

【結 果】

```
MMMMMMMMMMMMMMMMMMMMMMMM
```

(134) 【プログラム例】

```
DIM a$(5), s$(5) —— 社名が a$, 初任給にあう*の文字列が s$ に入る
FOR i = 1 TO 5 —— データを読んで社名を a$, 初任給にあう*の文字列を s$ に代入
    READ a$(i), x
    s$(i) = STRING$(x, "*")
NEXT i
b$ = "+" + STRING$(9, "-") —— b$ に "+-----" を代入
LOCATE 3, 20: PRINT "5ｼﾔ / ﾖﾆﾝｷｭｳ(ﾏﾝｴﾝ)" —— 標題
FOR i = 1 TO 5 —— 社名と+と初任給にあう*を表示
    LOCATE 3 + 2 * i, 5: PRINT a$(i)
    LOCATE 3 + 2 * i, 16: PRINT "+"
    LOCATE 3 + 2 * i, 17: PRINT s$(i)
NEXT i
```

```

FOR i = 1 TO 3
    LOCATE 15, 16 + (i - 1) * 10: PRINT b$
NEXT i
LOCATE 15, 16 + (i - 1) * 10: PRINT "+"
FOR i = 0 TO 30 STEP 10
    LOCATE 16, 15 + i: PRINT i
NEXT i
END
DATA ナカニッソ, 15, ニホンテツコウ, 19, トウキョウSS, 22
DATA シマシヨウジ, 12, ホンダ インサツ, 10

```

軸の表示

初任給額を表示

【結 果】

```

          シヤ ノ シヨニンキュウ(マンエン)

ナカニッソ *****
ニホンテツコウ *****
トウキョウSS *****
シマシヨウジ *****
ホンダ インサツ *****

+-----+-----+-----+
0          10         20         30

```

(135) 【プログラム例】

```

a$ = "Microsoft"
b = LEN(a$)
FOR i = 1 TO b
    PRINT MID$(a$, i, 1) + SPACES$(2);
NEXT i
PRINT
END

```

文字列を a\$ に代入

b に文字数を代入

1 文字ずつ後に 2 つの空白をつけて表示

【結 果】

M i c r o s o f t

(136) 【プログラム例】

```

a$ = "USA"
b$ = SPACES$(3)
FOR i = 1 TO 77
    LOCATE 5, i: PRINT a$
    FOR j = 1 TO 500: NEXT j
    LOCATE 5, i: PRINT b$
NEXT i
END

```

文字列を a\$ に代入

3 空白の文字列を b\$ に代入

5 行 1 列から 77 列まで 1 列ごとに USA を表示し、それを消去する

【結 果】

USA (USA が左から右へ動きます)

17章 漢 字

LEFT\$, RIGHT\$, KMID\$, KMID\$=, KLEN, KINSTR, KEXT\$, JIS\$,
CHR\$, KTN\$

【例題 123】 漢字の表示

”クイックベシク を 学びましょう。”を表示するプログラムをつくれ。

【解 説】 ◎ 漢字を表示します。

- ① 次の形で漢字を表示します。

```
PRINT "クイックベシク を 学びましょう。"
```

これは次のようにしても同じです。

```
a$="クイックベシク を 学びましょう。"
```

```
PRINT a$
```

- ② 漢字は漢字フロントエンドプロセッサを使います。本例は”ATOK6”を使用しています。フロントエンドプロセッサの組込みは準備編のとおりです。また、漢字変換については”ATOK6”を使用しているワープロソフトの説明書によってください。ここでは次の点についてのみふれておきます。

- ③ **CTRL** と **XFER** キーをおします。漢字変換モードとなります。 **f・10** キーをおすことで次の5つのモードが選べます。

連ローマ字漢字	ローマ字で入力して漢字変換します
連カナ漢字	カナで入力して漢字変換します
半角英数カナ	半角の英、数、カナがそのまま入ります
コード JIS	漢字 JIS コードで漢字を入力します
記号	記号で入力します

- ④ いずれも終了は **CTRL** + **XFER** です。

- ⑤ 本例についてのキー操作を示します。

PRINT”の後 **CNTR** + **XFER** をおします。 **f・10** キーを4回おして、記号モードとします。

↑ **↓** キーをおして、”ク”を含む行を表示させます。 ”ク”が反転しています。 **↓** します。同様の手順で矢印キーと **↓** キーで”イックベシク”まで入力します。 **↓** キーをおして空白をつくり **↓** します。 **f・10** キーで連ローマ字漢字モードとし、 **W** **O** とします。 **↓** キーをおし **↓** します。 ”を”が表示されます。 **M** **A** **N** **A** **B** **I** **M** **A** **S** **H** **O** **U** とキーをおした後、 **XFER** をおします。 ”学びましょう”となります。 **↓** で確定します。 **f・10** キーを3回おしてコード JIS モードとして、2123 **↓** します (2123 は ”。”の JIS コードです)。

最後に **CTRL** + **XFER** で漢字変換モードは終了です。

”を記入して、PRINT 命令ができました。

【プログラム例】

```
PRINT "クイックベーシック を 学びましょう。"
END
```

【結 果】

クイックベーシック を 学びましょう。

【例題 124】 漢字文字列の左側の文字のとり出し

”クイックベーシック を 学びましょう。”の中から”クイック”をとり出して表示するプログラムをつくれ。

【解 説】 ◎ 漢字文字列の中から左側の文字列をとり出します。

- ① a\$を”クイックベーシック を 学びましょう。”とします。
- ② LEFT\$はa\$の中から左側の指定した文字列をとり出します。
- ③ 漢字の場合は1文字が2バイトです。したがって、次のように8の指定で4文字をとり出します。

クイックベーシック を 学びましょう。

LEFT\$(a\$,8)

左側4文字

【プログラム例】

```
a$ = "クイックベーシック を 学びましょう。"
b$ = LEFT$(a$, 8)
PRINT b$
END
```

【結 果】

クイック

【例題 125】 漢字文字列の右側の文字列

”クイックベーシック を 学びましょう。”の中から右側7文字をとり出して表示するプログラムをつくれ。

【解 説】 ◎ 漢字文字列の右側7文字をとり出します。

- ① a\$を”クイックベーシック を 学びましょう。”とします。
- ② 次の形で右側7文字をとり出します。

RIGHT\$ (a\$,14)

右側の文字列

7文字. 漢字では1文字を2と数える
元の文字列

【プログラム例】

```
a$ = "クイックベーシック を 学びましょう。"
b$ = RIGHT$(a$, 14)
PRINT b$
END
```

【結 果】

学びましょう。

【例題 126】 漢字文字列の中間の文字列

“クイックベーシック を 学びましょう。”の中から 5 文字目から 5 文字を取り出して表示するプログラムをつくれ。

【解 説】 ◎ 漢字文字列の中間の文字列を取り出します。

- ① a\$と“クイックベーシック を 学びましょう。”とします。
- ② 次の形で中間の文字列を取り出します。

KMID\$
(a\$, 5, 5)

中間の文字列

元の文字列

5 文字

5 文字目から

KMID\$では漢字 1 文字を 1 と数える

この例では“ベーシック”となります。

【プログラム例】

```
a$ = "クイックベーシック を 学びましょう。"
b$ = KMID$(a$, 5, 5)
PRINT b$
END
```

【結 果】

ベーシック

【例題 127】 文字列の長さ

“クイックベーシック を 学びましょう。”を 1 文字ずつ取り出して表示するプログラムをつくれ。

【解 説】 ◎ 文字列の長さを求めます。

- ① a\$に“クイックベーシック を 学びましょう。”を代入します。
- ③ KLEN(a\$)は文字列 a\$の長さ、文字数を求めます。KLEN は漢字を 1 文字を 1 と数えます。

【プログラム例】

```
a$ = "クイックベーシック を 学びましょう。"
b = KLEN(a$)
FOR i = 1 TO b
```

```

PRINT KMID$(a$, i, 1);
NEXT i
PRINT
END

```

【結 果】

クイックベーシック を 学びましょう。

【例題 128】 文字列の入れかえ

a\$を”平成元年 12 月 31 日”とし、b\$に”11 月 15 日”を代入し、”12 月 31 日を”11 月 15 日”で入れかえて表示するプログラムをつくれ。

【解 説】 ◎ 漢字文字列の 1 部の入れかえをします。

- ① a\$を”平成元年 12 月 31 日”とします。b\$を”11 月 15 日”とします。
- ② ”12 月 31 日”の部分を”11 月 15 日”と入れかえます。

平成元年 12 月 31 日

↑

11 月 15 日

KMID\$(a\$, 5, 6)=b\$

a\$の 5 番目から 6 文字を b\$で入れかえ

【プログラム例】

```

a$ = "平成元年 1 2 月 3 1 日"
b$ = " 1 1 月 1 5 日"
KMID$(a$, 5, 6) = b$
PRINT a$
END

```

【結 果】

平成元年 1 1 月 1 5 日

【例題 129】 検 索

a\$を”クイックベーシック を 学びましょう。”, b\$を”学びましょう”, c\$を”クイック C”として、a\$の最初から b\$, c\$があるかどうかを検索するプログラムをつくれ。

【解 説】 ◎ 漢字文字列の検索をします。

- ① a\$を”クイックベーシック を 学びましょう。”, b\$を”学びましょう”, c\$を”クイック C”とします。
- ② KINSTR(1,a\$,b\$)は a\$の中に b\$があるかどうかを検索します。このとき 1 番目からさがします。KINSTR は漢字 1 文字を 1 と数えて検索します。見つかった場合、それが何番目からあるか、その番号を与えます。”クイックベーシック を 学びましょう。”の中に”学びましょう”は 13 番

目からありますので INSTR(1,a\$,b\$)は13です。見つからないときは0です。

【プログラム例】

```
a$ = "クイックベ—シ—ック を 学びましょう。"
b$ = "学びましょう"
c$ = "クイック C"
PRINT INSTR(1, a$, b$)
PRINT INSTR(1, a$, c$)
END
```

【結 果】

13
0

【例題 130】 全角文字と半角文字

"A はアルファベットの最初の文字です"の半角文字と全角文字をわけて表示するプログラムをつくれ。ただし, "A"は半角文字, 他は全角文字とします。

【解 説】 ◎ 半角文字と全角文字を取り出します。

- ① KEXT\$(a\$,0)はa\$の中の半角文字を取り出します。この例では"A"のみです。
- ② KEXT\$(a\$,1)はa\$の中の全角文字を取り出します。

【プログラム例】

```
a$ = "Aはアルファベットの最初の文字です"
b$ = KEXT$(a$, 0)
c$ = KEXT$(a$, 1)
PRINT b$
PRINT c$
END
```

【結 果】

A
はアルファベットの最初の文字です

【例題 131】 漢字の JIS コード

"世界"の"世"の JIS コードを表示するプログラムをつくれ。

【解 説】 ◎ 漢字の JIS コードを求めます。

- ① JIS\$(a\$)では文字列 a\$の先頭文字が2 バイト文字のときその文字の16 進の JIS コードを4 バイトのASCII文字列で求めます。1 バイト文字のときはその文字の10 進の JIS コードを3 バイト以下のASCII文字列で求めます。
- ② a\$を"世界"とすると"世"の JIS コードで4024 です。
- ③ a\$を"A"とすると65, a\$を"a"とすると97 です。

【プログラム例】

```
a$ = "世界"
PRINT JIS$(a$)
END
```

【結 果】

4024

【例題 132】 漢字の CHR\$

CHR\$を使って"世界"と表示するプログラムをつくれ.

【解 説】 ◎ シフト JIS コードで漢字を表示します.

① CHR\$(3173)で"世", CHR\$(1953)で"界"を持ちます.

② 0 から 255 のときは 1 バイト文字, 256 から 11,535 のときはシフト JIS に対応する 2 バイト文字を与えます.

【プログラム例】

```
PRINT CHR$(3173);
PRINT CHR$(1953)
END
```

【結 果】

世界

【例題 133】 漢字文字列の句点コード

"ABCD 大文字 abcd 小文字"の各文字の句点コードを表示するプログラムをつくれ.

【解 説】 ◎ 句点コードを求めます.

① KTN\$(c\$)は文字 c\$が 2 バイト文字のときその文字に対応する句点コードを持ちます. 1 バイト文字のときはアスキーコードを持ちます. "A"は 65, "B"は 66, "大"は 3471 などです.

【プログラム例】

```
a$ = "ABCD大文字abcd小文字"
b = KLEN(a$)
FOR i = 1 TO b
  c$ = KMID$(a$, i, 1)
  PRINT KTN$(c$); " ";
NEXT i
PRINT
END
```

【結 果】

65 66 67 68 3471 4224 2790 97 98 99 100 3014 4224 2790

【例題 134】 漢字の入力

漢字文字列を入力して、表示するプログラムをつくれ。

【解 説】 ◎ 漢字の入力をします。

- ① 漢字の入力も、普通の文字列の入力と同様に INPUT x\$で行います。
- ② プログラムが実行されると?を表示し、入力を要求しますので、**CTRL** キーと **XFER** キーをおして漢字入力モードとして、漢字を入力します。
- ③ 入力が決まったら **CTRL** キーと **XFER** キーを再度おして、漢字入力モードを解除します。

【プログラム例】

```
INPUT x$
PRINT x$
END
```

【結 果】

```
? お お さ か
お お さ か
? 東 京
東 京
```

〔演習問題〕

- (137) "平成元年 12 月 31 日"と表示するプログラムをつくれ。
- (138) "平成元年 12 月 31 日"のうち元年をとり出して表示するプログラムをつくれ。
- (139) "クイックベシック を 学びましょう。"を a\$, "始めます。[]]"を b\$として、a\$の最後の 7 文字を b\$で入れかえて表示するプログラムをつくれ。
- (140) "たけやがやけた"を後ろから 1 文字ずつ表示するプログラムをつくれ。
- (141) "平成元年 12 月 31 日"の 1 文字ずつの句点コードを表示するプログラムをつくれ。
- (142) "ABCD 大文字 abcd 小文字"のうち半角文字と全角文字を各々とり出して表示するプログラムをつくれ。
- (143) 1666 から 2000 までの CHR\$の漢字を表示するプログラムをつくれ。
- (144) 820 から 852 までの CHR\$の文字を表示するプログラムをつくれ。
- (145) "クイックベシック"を a\$とし、a\$の JIS\$を表示するプログラムをつくれ。
- (146) a\$を"クイックベシック"として、1 文字おきに表示するプログラムをつくれ。

〔解 答〕

(137) 【プログラム例】

```
PRINT "平成元年 1 2 月 3 1 日"
END
```

【結 果】

平成元年 1 2 月 3 1 日

(138) 【プログラム例】

```

a$ = "平成元年 1 2 月 3 1 日"
b$ = KMID$(a$, 3, 2)
PRINT b$
END

```

【結 果】

元年

(139) 【プログラム例】

```

a$ = "クイックベ—シツク を 学びましょう。"
b$ = "始めます。"
KMID$(a$, 13, 7) = b$
PRINT a$
END

```

【結 果】

クイックベ—シツク を 始めます。

(140) 【プログラム例】

```

a$ = "たけやがやけた"
b = KLEN(a$)
FOR i = b TO 1 STEP -1
    PRINT KMID$(a$, i, 1);
NEXT i
PRINT
END

```

【結 果】

たけやがやけた

(141) 【プログラム例】

```

a$ = "平成元年 1 2 月 3 1 日"
b = KLEN(a$)
FOR i = 1 TO b
    c$ = KMID$(a$, i, 1)
    PRINT KTNS$(c$); " ";
NEXT i
PRINT
END

```

【結 果】

4231 3214 2421 3915 0317 0318 2378 0319 0317 3892

(142) 【プログラム例】

```
a$ = "ABCD大文字abcd小文字"
b$ = KEXT$(a$, 0)
c$ = KEXT$(a$, 1)
PRINT b$
PRINT c$
END
```

【結 果】

```
ABCDabcd
大文字小文字
```

(143) 【プログラム例】

```
FOR i = 1666 TO 2000
  a$ = CHR$(i)
  PRINT a$;
NEXT i
END
```

【結 果】

鞍稲浦奄岡稼戒各
 闇逸既黃禾懷嚇
 案溢姥園國禍恢割
 暗老縵內驚珂悔鈎
 按一蔚厥澳火怪
 庵榎翁河快柿
 安郁頃閱王歌迴垣
 拾育嚙越毆架壞娃
 粟域渦謁欧果塊馨
 或亥白悅橫暇回湮
 鮎井驢馱旺科鰓骸
 綾医丑益押寡会鎧
 鈎遺窺疫忝家介該
 飴違鷄液往嫁駕街
 虻謂卵銳奧夏餓蓋
 姐衣雨詠央嘉雅碍
 宛葵迂衛凹可賀涯
 扱胃羽英甥加娥概
 幹緯烏穎汚佳茅慨
 压維宇於偈臥崖
 梓移右盈塩伽画害
 鏐異吋瑛璽何牙咳
 芦畏韻洩鉛飯我外
 葦為隱泳遠化峨効
 旭椅陰永蓮下俄凱
 渥易院采范音貝
 握慰薩曳穩霞階郭
 惡意胤映緣温過開較
 恠淫影猿恩迦蟹赫
 茜尉飲嬰燕卸貨芥角
 葵威引宮煙簪繪覺
 逢委姻穀焰乙課皆穫
 始夷因餌炎牡蝦界
 挨■荏演桶■灰獲
 愛偉咽雲浴臆華海殺
 哀依印運援憶荷械核
 阿位允云掩屋茄晦格
 娃伊銷嚙怨億苛魁攪
 啞以芋閨延荻花改拈
 亞杏茨瓜宴冲箇拐廓

(144) 【プログラム例】

```
FOR i = 820 TO 852
  a$ = CHR$(i)
  PRINT a$;
NEXT i
END
```

【結 果】

А В В Г Д Е Е Ж З И Й К Л М Н О П Р С Т У Ф Х Ц Ч Ш Щ Ъ Ы Ь Э Ю Я

(145) 【プログラム例】

```
a$ = "クイックベ—シック"
PRINT JIS$(a$)
END
```

【結 果】

252F

(146) 【プログラム例】

```
a$ = "クイックベ—シク"  
b = KLEN(a$)  
FOR i = 1 TO b STEP 2  
    PRINT KMID$(a$, i, 1);  
NEXT i  
PRINT  
END
```

【結 果】

ク ッ ベ シ ク

18章 桁・行の指定

LOCATE

【例題 135】 LOCATE で表示位置を指定

1 行 1 桁から "abc", 2 行 1 桁から "def", 3 行 10 桁から "ghi" 5 行 13 桁から "jkl" を表示するプログラムをつくれ.

【解 説】 ◎ LOCATE で表示開始位置を決めます.

- ① LOCATE x, y によって表示開始位置を決めます. x が行, y が桁を示します. 下図のように画面左上を 1 行 1 桁とします. BASIC では通常, 左上を 0 桁 0 行とし, x を桁, y を行で表わしますので注意してください.

Quick BASIC

```
(1,1) (1,2) (1,3).....
(2,1)
(3,1)
⋮
```

Nee BASIC

```
(0,0) (1,0) (2,0).....
(0,1)
(0,2)
⋮
```

左上を 1 行 1 桁とする. 行, 桁の順に示す. 左上を 0 桁 0 行とする. 桁, 行の順に示す.

- ② LOCATE 1, 1 では 1 行 1 桁にカーソルを移します. 続いて, PRINT "abc"により, 1 行 1 桁から "abc"を表示します.
- ③ LOCATE 2, 1 では 2 行 1 桁にカーソルを移します. 続いて, PRINT "def"で, 2 行 1 桁から "def"を表示します.
- ④ LOCATE, 10 では行を変更せず 10 桁へカーソルを移します. すなわち項③の PRINT "def"で改行されていますから, 実際は 3 行目, 10 桁にカーソルがきます. そこから "ghi"を表示します.
- ⑤ LOCATE 5 では 5 行目の桁は変更しない位置へカーソルを移します. 実際は 3 行目, 10 桁目から "ghi";を表示していますのでカーソルは 3 行目 13 桁目にあります. したがって, 桁は同じですから 5 行目の 13 桁目にカーソルを移します. そこから PRINT "jkl"で表示をします.

【プログラム例】

```
LOCATE 1, 1 — 1 行 1 桁から abc を表示
PRINT "abc" —
LOCATE 2, 1 — 2 行 1 桁から def を表示
PRINT "def" —
LOCATE , 10 — 3 行 10 桁から ghi を表示
PRINT "ghi"; —
```

```
LOCATE 5 _____ 5行13桁から jkl を表示
PRINT "jkl" _____
END
```

【結 果】

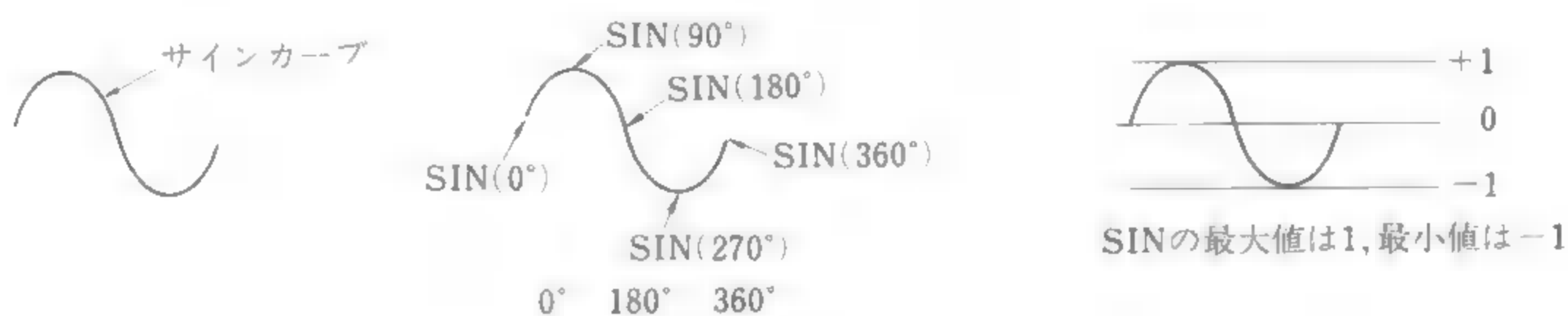
```
abc
def
      ghi
            jkl
```

【例題 136】 サインカーブを描く

"0"を使ってサインカーブをかくプログラムをつくれ。

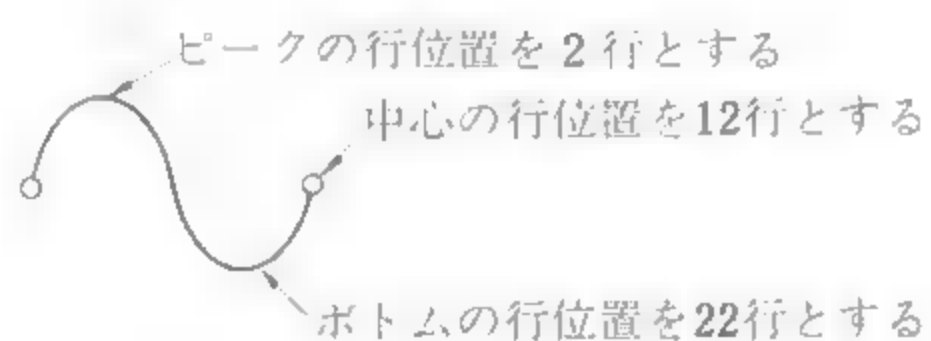
【解 説】 ◎ LOCATE を使ってサインカーブを描きます。

- ① サインカーブは 0 度から 360 度までの SIN の値をプロットしたものです。



- ② キャラクタ"0"を LOCATE を使って表示します。次のように考えます。

[行位置の決め方]



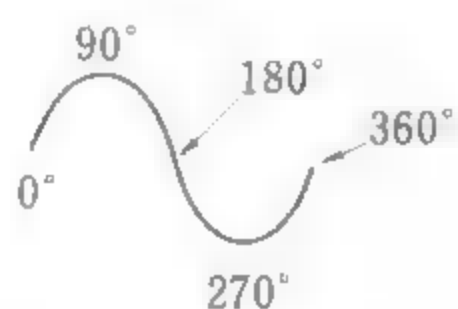
中心を 12 行, SIN の最大値 1 のとき 2 行目

SIN の最小値 -1 のとき 22 行目とする。

したがって、行位置は $12 - 10\text{SIN}(0 \sim 360^\circ)$ となります。

0 ~ 360° は適当な間隔とします。桁位置の決定のところで説明します。

[桁位置の決め方]



0° のときの桁位置を 4 桁, 90° のときの桁位置を 22 桁

180° の桁位置を 40 桁, 270° の桁位置を 58 桁

360° の桁位置を 76 桁とします。

表示は 0°, 5°, 10°, 15°, ... 355°, 360° の場合の桁位置のときとします。

したがって、4 桁目, 5 桁目, 6 桁目, ... 76 桁目に表示されます。

- ③ したがって、FOR i=0 TO 360 STEP 5 のくり返しで、i 度の際の行位置と桁位置を求めると次のようになります。

$12 - 10 * \text{SIN}(i / 180 * 3.14159)$ 行位置

$(i + 20) / 5$ 桁位置

- ④ したがって、LOCATE i, (i+20)/5 によって"0"を表示する位置を決めます。

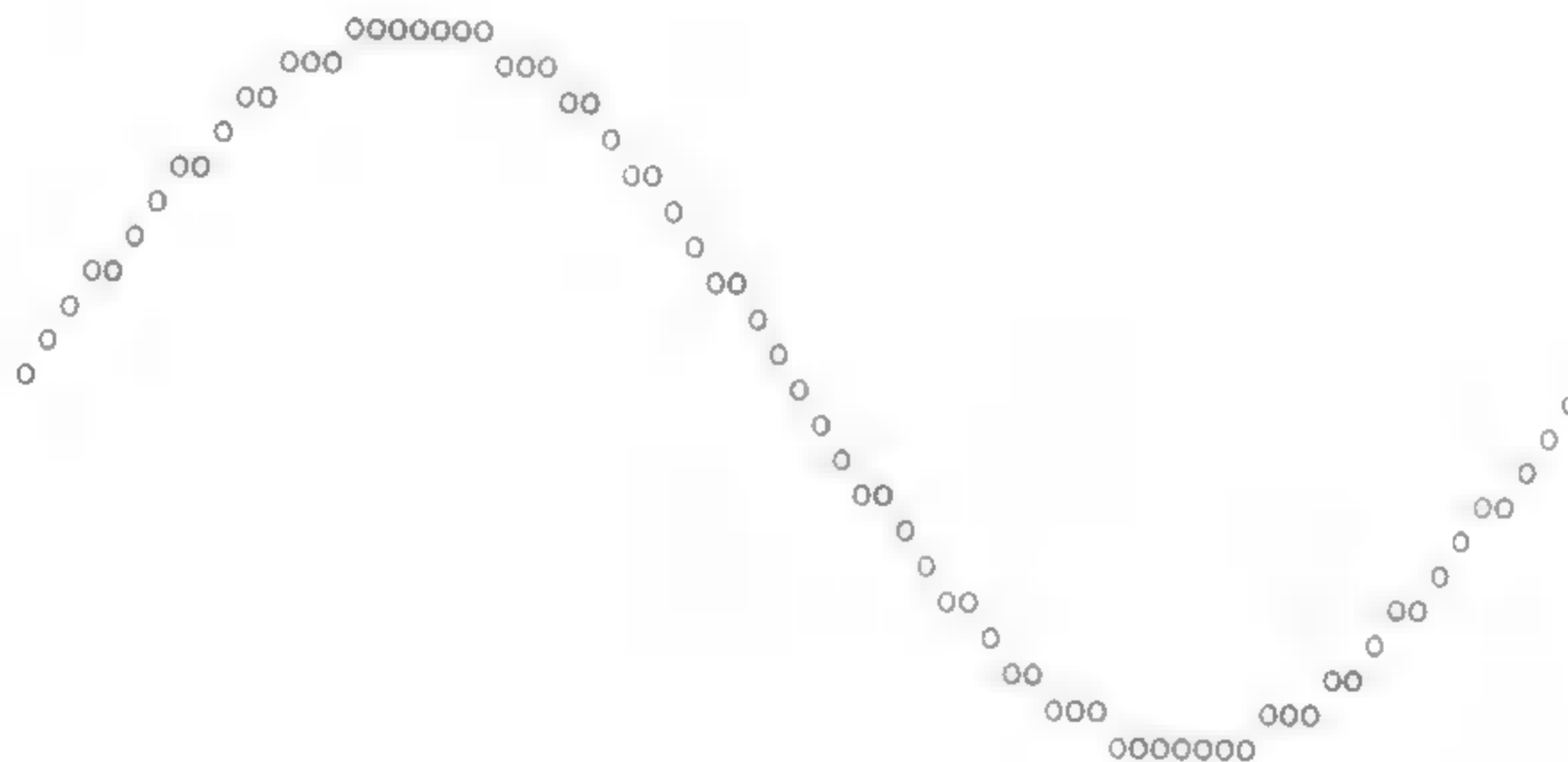
【プログラム例】

```

FOR i = 0 TO 360 STEP 5 ————— 0～360°まで5°ごとにくり返し
  j = 12 - 10 * SIN(i / 180 * 3.14159) ———— 行の計算
  LOCATE j, (i + 20) / 5 ————— j 行, (i+20)/5 桁を指定
  PRINT "O" ————— O の表示
NEXT i
END

```

【結 果】

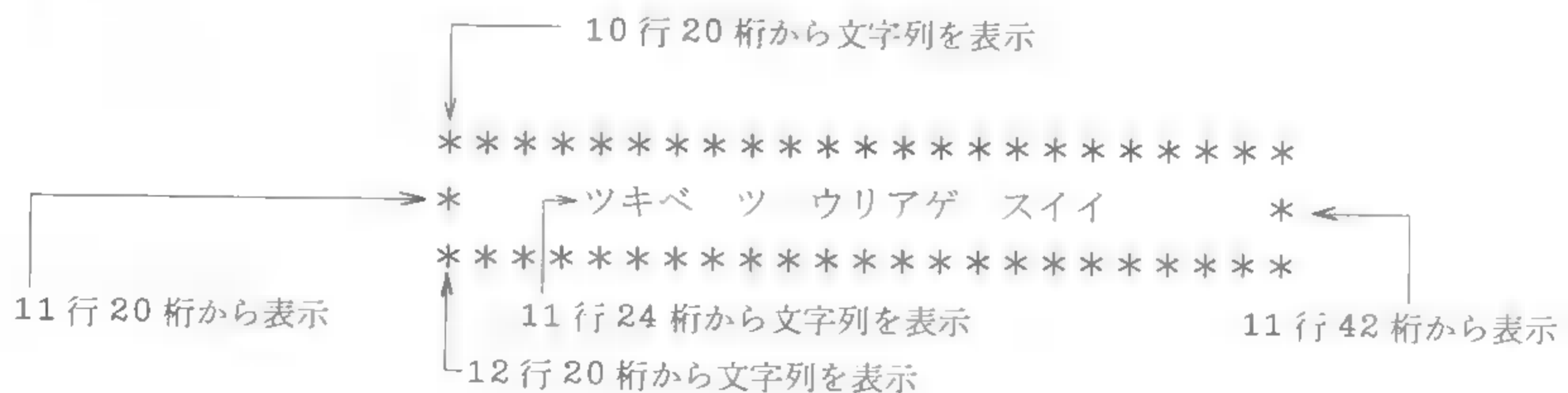


【例題 137】 標題を表示

結果の標題を左上を 10 行 20 桁として表示するプログラムをつくれ.

【解 説】 ◎ 標題を指定位置へ表示します.

- ① "＊"を 23 個並べた文字列を a\$, "＊"1 つを b\$, 標題の文字列を c\$に代入します. 文字列は全体で 15 桁ですから, 前後に 3 桁ずつ空白をつけ, またその前後に"＊"を 1 つずつつけます. したがって, 全体は 23 桁です. したがって, 上下の行に表示する"＊"の文字列は 23 個の"＊"を持たせます.
- ② 次のように LOCATE 文で位置を指定して表示します.



【プログラム例】

```

READ a$, b$, c$ ————— データ文の文字列を代入
LOCATE 10, 20 ————— a$の表示
PRINT a$
LOCATE 11, 20 ————— b$の表示
PRINT b$

```

```

LOCATE 11, 24 ————— c$の表示
PRINT c$ —————
LOCATE 11, 42 ————— b$の表示
PRINT b$ —————
LOCATE 12, 20 ————— a$の表示
PRINT a$ —————
END
DATA "*****"
DATA "*"
DATA "ツキヘツ ウリアケ スイ"

```

【結 果】

```

*****
*   ツキヘツ ウリアケ スイ   *
*****

```

【例題 138】 位置指定で入力

2 行 5 桁目から文字を入力し、4 行 5 桁目から表示するプログラムをつくれ。

【解 説】 ◎ LOCATE は入力位置の指定もできます。

- ① LOCATE 2, 5 の後 INPUT x\$ とするとカーソルを 2 行 5 桁へ移動し、そこから x\$ の入力を行います。ただし、INPUT 文の? の位置が 5 桁目です。

? ——— LOCATE 2, 5 により INPUT 文の? は 2 行 5 桁目に表示される

【プログラム例】

```

LOCATE 2, 5 ——— 2 行 5 桁から x$ の入力
INPUT x$ ———
LOCATE 4, 5 ——— 4 行 5 桁から表示
PRINT x$ ———
END

```

【結 果】

? 入力待ち画面

? 123456 123456 を入力の例

123456

【例題 139】 文字の移動

A を画面左から右へ動かすプログラムをつくれ。

【解 説】 ◎ 文字を画面左から右へ動かします。

① 文字の移動は次のようにします。



A が左から右へ動く。

同じ行の桁を 1 から 80 までにすると 1 桁目から 80 桁目まで動く。

【プログラム例】

```
FOR i = 1 TO 80——— i を 1 から 80 まで、i は桁を示す
  LOCATE 3, i——— 3 行 i 桁に A を表示
  PRINT "A"———
  LOCATE 3, i——— 3 行 i 桁にスペースを表示
  PRINT " "———
NEXT i
END
```

【結 果】

A A が左から右へ動く

〔演習問題〕

(147) ドのデータを 1 行目～9 行目の 5 桁目から 1 つずつ表示するプログラムをつくれ。

データ： PC-9801, PC-8801, PC-98D0

IBM-XT, IBM-PS2, IBM-AT

FM-towns, Canon-AXi, Panacom-M530

(148) A を結果のように表示するプログラムをつくれ。これは "A" を 1 行 1 桁, 2 行 2 桁, 3 行 3 桁, …15 行 15 桁に表示したものである。

(149) "0" を使ってコサインカーブを表示するプログラムをつくれ。

〔解 答〕

(147) 【プログラム例】

```
FOR i = 1 TO 9——— 9 回くり返し
  READ a$——— 文字列を読む
  LOCATE i, 5——— i 行 5 桁から表示
  PRINT a$———
NEXT i
END
DATA PC-9801, PC-8801, PC-98D0
DATA IBM-XT, IBM-PS2, IBM-AT
DATA FM-towns, Canon-AXi, Panacom-M530
```

【結 果】

PC-9801
PC-8801
PC-98DO
IBM-XT
IBM-PS2
IBM-AT
FM-towns
Canon-AXi
Panacom-M530

(148) 【プログラム例】

```
FOR i = 1 TO 15
  LOCATE i, i      i 行 i 桁から "A" を表示
  PRINT "A"
NEXT i
END
```

【結 果】

(149) 【プログラム例】

```
FOR i = 0 TO 360 STEP 5
  j = 12 - 10 * COS(i / 180 * 3.14159) —— 行位置を求める
  LOCATE j, (i + 20) / 5 —— j 行, (i+20)/5 桁に o を表示
  PRINT "o"
NEXT i
END
```

【結 果】

A scatter plot showing a parabolic trend. The data points are represented by open circles, and a shaded gray area indicates the confidence interval. The curve is symmetric, opening upwards, with the minimum value at the center of the plot.

19 章 タイマー割り込み・キー割り込み・サブルーチンへ分岐

TIMER ON/OFF/STOP, ON TIMER GOSUB~RETURN
KEY() ON/OFF/STOP, ON KEY() GOSUB~RETURN

【例題 140】 タイマー割り込み

FOR i=1 TO 30000:NEXT i をくり返す中で、1 秒ごとに割り込みでサブプログラム count へ分岐し、カウンタを 1 ずつ増やし表示するプログラムをつくれ。

【解 説】 ㉓ タイマー割り込みをします。

- ① 次の形でくり返しの途中で 1 秒ごとに割り込みがかかり、サブプログラム count へ分岐します。

```
TIMER ON                      割り込みをオンとする
|
ON TIMER(1) GOSUB count
|
1 秒ごとに割り込みサブプログラム count へ分岐
```

FOR i=1 TO 30000:NEXT i

- ② 割り込みの開始は TIMER ON です。割り込みの停止は TIMER OFF、割り込みの中断は TIMER STOP です。
- ③ サブルーチンへの分岐は GOSUB プログラム名です。プログラム名はラベルのように使います。サブルーチンの終了は RETURN です。

```
GOSUB count
/
count:
/
RETURN
```

count の名前のサブルーチンへ分岐

RETURN で GOSUB の次へ戻る

- ④ このプログラムはほとんどが FOR i=1 TO 30000:NEXT i の処理に時間を使います。その間 1 秒ごとに割り込みが入り、サブプログラム count へ分岐して、c に 1 を加え表示していきます。このプログラムの処理に約 16 秒かかることがわかります。

【プログラム例】

```
TIMER ON ————— 割り込み開始
ON TIMER(1) GOSUB count ————— 1 秒ごとに割り込みでサブプログラム count へ分岐
```

```

FOR i = 1 TO 30000: NEXT i —— くり返し
PRINT
END
count: _____
      c = c + 1
      PRINT c;
RETURN _____

```

サブプログラム
c に 1 を加えて表示

【結 果】

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

【例題 141】 キー割り込み

ファンクションキー1, 2, 3 がおされたら, 割り込みで"キー1 ガ オサレマシタ", "キー2 ガ オサレマシタ", "キー3 ガ オサレマシタ"と表示して終了するプログラムをつくれ. ただし, キー3 がおされたら表示の後終了するものとする.

【解 説】 ① ファンクションキーによる割り込みをします.

- ① ファンクションキー1~10 を押すことで割り込みをさせることができます.
- ② KEY(1) ON と ON KEY(1) GOSUB a によってファンクションキー1 がおされると割り込みでサブプログラム a へ分岐します.
- ③ KEY(1) の 1 の部分が 1 から 10 を使いファンクションキー1~10 を示します.
- ④ KEY() OFF は割り込みの禁止, KEY() STOP は割り込みを中断します.
- ⑤ KEY 割り込みが ON のときはファンクションキーに割りあてられているファンクションは無効です.

【プログラム例】

```

KEY(1) ON _____ ファンクションキー1~3 の割り込みをオン
KEY(2) ON _____
KEY(3) ON _____
ON KEY(1) GOSUB a _____ ファンクションキー1~3 からおされると各々
ON KEY(2) GOSUB b _____ サブプログラム a, b, c へ分岐
ON KEY(3) GOSUB c _____
WHILE 1 _____ 無限ループ
  Z = Z
WEND _____
e:
END
a: _____ サブプログラム a
  PRINT "キー1 ガ オサレマシタ"
RETURN _____
b: _____ サブプログラム b
  PRINT "キー2 ガ オサレマシタ"
RETURN _____
c: _____ サブプログラム c
  PRINT "キー3 ガ オサレマシタ"
RETURN e _____

```

【結 果】





キ-1 カ` オサレマシタ
 キ-2 カ` オサレマシタ
 キ-2 カ` オサレマシタ
 キ-1 カ` オサレマシタ
 キ-3 カ` オサレマシタ

ファンクションキー
 1, 2, 2, 1, 3 がおされた例

【例題 142】 矢印キーによる割り込み

矢印キーがおされたらその方向の矢印を表示するプログラムをつくれ。





【解 説】 ◎ 矢印キーによる割り込みをします。

- ①  キーは 11,  キーは 12,  キーは 13,  キーは 14 によって割り込みができます。
 使い方は【例題 141】と同じです。

【プログラム例】

```

KEY(11) ON _____
KEY(12) ON _____
KEY(13) ON _____
KEY(14) ON _____
ON KEY(11) GOSUB a
ON KEY(12) GOSUB b
ON KEY(13) GOSUB c
ON KEY(14) GOSUB d
WHILE 1
  Z = Z
WEND
e:
END
a:
  PRINT "↑"
  RETURN e
b:
  PRINT "←"
  RETURN e
c:
  PRINT "→"
  RETURN e
d:
  PRINT "↓"
  RETURN e
  
```

キー11～14の割り込みオン
 11は, 12は, 13は, 14は キー

キー11～14による割り込みによってサブプログラムa～dへ分岐

無限ループ

サブプログラムa

サブプログラムb

サブプログラムc

サブプログラムd

【結 果】

↑
 →
 ↓
 ←
 ↓

, , , ,  を入力の場合

【例題 143】 ユーザ一定義キー割り込み

【例題 11-1】

A キー, % キー, CTRL キーと C キーがおされたら各々 A, %, end と表示するプログラムをつくれ.

【解 説】 ◎ ユーザー定義キーを使ってキー割り込みをします。

- ① キー番号 15 から 25 はユーザーが定義して使います。
- ② 定義の仕方は次のとおりです。

KEY n , CHR\$(&Hx)+CHR\$(&Hy)

↑ ↑ ↑

15 から 25 まで キーボードスキャンコードで、

&H00 はキーボードフラッグなし [ESC] が 00, 1! が 01,

&H01 は [SHIFT] キー [2"] が 02...

&H02 は [CAPS] キー [CTRL] が 74 のようになっています。

&H04 は [カナ] キー

&H08 は [GRPH] キー

&H10 は [CTRL] キー

- ③ たとえば、次のようになります。

KEY 15,CHR\$(&H00)+CHR\$(&H1D)	A キーを 15 に割りあて
KEY 16,CHR\$(&H01)+CHR\$(&H05)	[SHIFT] キーと [5^x] キーを 16 に割りあて
KEY 17,CHR\$(&H10)+CHR\$(&H2B)	[CTRL] キーと C キーを 17 に割りあて

【プログラム例】

```

KEY 15, CHR$(&H0) + CHR$(&H1D) ———— キー15～17 の定義
KEY 16, CHR$(&H1) + CHR$(&H5)
KEY 17, CHR$(&H10) + CHR$(&H2B) ————
KEY(15) ON ———— キー15～17 のオン
KEY(16) ON
KEY(17) ON ————
ON KEY(15) GOSUB a ———— キー割り込み
ON KEY(16) GOSUB b
ON KEY(17) GOSUB c ————
WHILE 1 ———— 無限ループ
    Z = Z
WEND ————
e:
END
a: ———— サブプログラム a
    PRINT "A"
RETURN e ————
b: ———— サブプログラム b
    PRINT "%"
RETURN e ————

```



```

c: PRINT "end"
   RETURN e

```

サブプログラム c

【結 果】

A A, %, CTRL + C を入力の場合
 %
 end

【例題 144】 割り込みを使ってプログラムの中断を防止

STOP キー, ESC キー, CTRL + C キーを各々 15, 16, 17 にキー定義して, 1 から 1000 までを表示するプログラムの中でこれらがおされてもプログラムが中断しないプログラムをつくれ.

【解 説】 ◎ キー割り込みを利用してプログラムの中断を防ぎます.

- ① STOP キーをプログラム実行中におすとプログラムが中断します. これを防ぐためにはキー割り込みを利用して, STOP キーがおされたら, サブプログラムへ分岐して, 単に RETURN すれば, プログラムは中断しません.
- ② 同様に ESC キー, CTRL キー + C キーを定義してキー割り込みを利用してプログラムの中断を防ぎます.

【プログラム例】

```

KEY 15, CHR$(&H0) + CHR$(&H60)
KEY 16, CHR$(&H0) + CHR$(&H0)
KEY 17, CHR$(&H10) + CHR$(&H2B)
KEY(15) ON
KEY(16) ON
KEY(17) ON
ON KEY(15) GOSUB a
ON KEY(16) GOSUB b
ON KEY(17) GOSUB c
FOR i = 1 TO 1000
  PRINT i;
NEXT i
PRINT
END
a: RETURN
b: RETURN
c: RETURN

```

STOP キーを 15, ESC キーを 16, CTRL
 キー + C キーを 17 で定義

キー 15~17 のオン

キー割り込み

1~1000 の表示

サブプログラム a~c, いずれも単に RETURN のみ,
 プログラムの中断を防ぐ

【結 果】 前半省略 (ESC キー, STOP キー, CTRL + C キーを途中でおす)

8	919	920	921	922	923	924	925	926	927	928	929	930	931	932	933	93
4	935	936	937	938	939	940	941	942	943	944	945	946	947	948	949	95
0	951	952	953	954	955	956	957	958	959	960	961	962	963	964	965	96
6	967	968	969	970	971	972	973	974	975	976	977	978	979	980	981	98
2	983	984	985	986	987	988	989	990	991	992	993	994	995	996	997	99
8	999	1000														

〔演習問題〕

- (150) 10 秒たったら, "10 ビョウ ケイカ" と表示して終わるプログラムをつくれ.
- (151) A, B, C キーがおされたらその数を数えて表示するプログラムをつくれ. ただし, どれかが 5 回となったら表示して終了するものとする.

〔解 答〕

(150) 【プログラム例】

```

TIMER ON ————— タイマ割り込み開始
ON TIMER(10) GOSUB count ——— 10 秒たったら count へ分岐
WHILE 1 ————— 無限ループ
    Z = Z
WEND
e:
END
count: ————— サブプログラム count
    PRINT "10 ビョウ ケイカ"
RETURN e

```

【結 果】

10 ビョウ ケイカ

(151) 【プログラム例】

```

KEY 15, CHR$(&H0) + CHR$(&H1D) ————— A, B, C キーを定義
KEY 16, CHR$(&H0) + CHR$(&H2D)
KEY 17, CHR$(&H0) + CHR$(&H2B)
KEY(15) ON ————— キー15~16 オン
KEY(16) ON
KEY(17) ON
ON KEY(15) GOSUB a ————— キー割り込み
ON KEY(16) GOSUB b
ON KEY(17) GOSUB c
WHILE 1 ————— 無限ループ
    Z = Z
WEND
e:
PRINT "A="; a; "がI " ————— 表示
PRINT "B="; b; "がI "
PRINT "C="; c; "がI "
END
a: ————— サブプログラム a
    a = a + 1
    IF a <= 4 THEN RETURN ELSE RETURN e ——— A の回数のカウント

```

b: ————— サブプログラム b
 b = b + 1 B の回数のカウント
 IF b <= 4 THEN RETURN ELSE RETURN e —————

c: ————— サブプログラム c
 c = c + 1 C の回数のカウント
 IF c <= 4 THEN RETURN ELSE RETURN e —————

【結 果】

A= 3 カイ (B A B B A C C A C B B の例)
 B= 5 カイ
 C= 3 カイ

20 章 シーケンシャルファイル

OPEN~FOR OUTPUT, OPEN~FOR INPUT, OPEN~FOR APPEND,
WRITE#, INPUT#, PRINT#, CLOSE#, EOF

【例題 145】 シーケンシャルファイルの書き込み

下のデータをシーケンシャルファイル `meibo.dat` にかくプログラムをつくれ。

データ：オオヤマタロウ, 38, 1570, ウエダヨシオ, 41, 1300
 クリタサブロウ, 25, 820, トクガワヨシオ, 52, 2300
 カワカミタロウ, 21, 420, アマダヨシミツ, 37, 1210
(名前, 年齢, 年収の順のデータとする)

【解 説】 ① シーケンシャルファイルをつくり書き込みをします。

- ① シーケンシャルファイルはテープと同じように前から順にデータを書いていきます。途中から書くことはできません。
- ② まずファイルをオープンします。ファイルのオープンは今後使用するファイルを明確にし、それに番号を与えて、今後その番号で識別していきます。シーケンシャルファイルの書き込みは OUTPUT を指定します。

```
OPEN      "meibo.dat" FOR OUTPUT AS#1
  ↑        ↑          ↑          ↑
  ファイルの ファイル名 書き込み用 オープン番号
オープン
```

この中でユーザーが変えられるのはファイル名の `meibo.dat` の部分とオープン番号の `1` のみです。1 から 255 が使えます。あとは書式のとおりとします。

- ③ 配列 `a$` と `x%` に読み込まれたデータを次の形でファイルへ書き込みます。書き込みは `WRITE#` を使います。ファイルの識別をオープン番号 `1` で識別します。

```
WRITE #1 a$(i),x%(i,1),x%(i,2)
  ↑      ↑      ↑      ↑      ↑
  ファイルへ 識別 名前 年齢 年収
  の書き込み 番号 1
```

- ④ 書き込みが終わったらオープン番号と同じ番号を指定してファイルをクローズします。

```
CLOSE #1
  ↑
  ファイルの 識別番号
  クローズ      #1 を省略するとすべてのファイルがクローズされます。
```

CLOSE がなくても END などによって自動的にクローズされます。

- ⑤ シーケンシャルファイルでは本例のように、文字データ、整数型データ、整数型データを1組として、この順で繰り返されます。このとき、文字データの長さが同じでなくてもよいところに大きな特長があります。

したがって、このファイルが何バイト使用しているのかデータの数だけではわかりません。

- ⑥ FOR OUTPUT を指定してファイルをオープンすると、同じ名前のファイルがなければ作成し、すでにあるときはそのデータをすべてなくしてしまいますから注意してください。

【プログラム例】

```

DIM a$(6), x%(6, 2)          配列の宣言
FOR i = 0 TO 5                データの読み込み
    READ a$(i), x%(i, 1), x%(i, 2)
NEXT i
OPEN "b:meibo.dat" FOR OUTPUT AS #1  ファイルのオープン
FOR i = 0 TO 5                ファイルヘータの書き込み
    WRITE #1, a$(i), x%(i, 1), x%(i, 2)
NEXT i
CLOSE #1                      ファイルのクローズ
END
DATA オオヤマタロウ, 38, 1570, ウエダ ヨシオ, 41, 1300
DATA クリタサブ`ロウ, 25, 820, トクカ`ワヨシオ, 52, 2300
DATA カワカミタロウ, 21, 420, アマタ`ヨシミツ, 37, 1210

```

【結 果】 データがディスクにかき込まれます。

MS-DOS の type コマンドでファイルの内容をみると次のようになっています。

```

type meibo.dat
"オオヤマタロウ", 38, 1570
"ウエダ`ヨシオ", 41, 1300
"クリタサブ`ロウ", 25, 820
"トクカ`ワヨシオ", 52, 2300
"カワカミタロウ", 21, 420
"アマタ`ヨシミツ", 37, 1210

```

【例題 146】 シーケンシャルファイルの読みだし

【例題 145】で書き込んだシーケンシャルファイル meibo.dat のデータを読み出して表示するプログラムをつくれ。

【解 説】 ㉠ シーケンシャルファイルを読み出します。

- ① シーケンシャルファイルはテープと同様に前から順にしか読み出せません。
- ② シーケンシャルファイルの読み出しは、書き込みと同様にまずファイルをオープンします。このとき、INPUT を指定します。

```

OPEN    "meibo.dat" FOR INPUT AS #2
ファイルの   ファイル名   読み出しの指定   オープン番号
オープン

```

- ③ 読み出しは次のようにします。読み出しは INPUT#を使います。

```
INPUT #2, a$, x%, y%
```

読み出された最初の文字列が a\$ に、次の 2 つの整数が x% と y% に代入されます。

- ④ 読み出しがすんだら、ファイルをクローズします。

```
CLOSE #2
```

クローズ ↑ 識別番号

【プログラム例】

```
OPEN "b:meibo.dat" FOR INPUT AS #2  ———— ファイルのオープン
FOR i = 1 TO 6  ————— データの読み出しと表示
    INPUT #2, a$, x%, y%
    PRINT a$; x%; y%
NEXT i
CLOSE #2  ————— ファイルのクローズ
END
```

【結 果】

```
オオヤマタロウ 38 1570
ウエタ`ヨシオ 41 1300
クリタサフ`ロウ 25 820
トクカ`ワヨシオ 52 2300
カワカミタロウ 21 420
アマタ`ヨシミツ 37 1210
```

【例題 147】 シーケンシャルファイルにデータを追加

【例題 145】のファイル meibo.dat に追加して下のデータを書き込むプログラムをつくれ。

データ：カドカワヒロシ, 35, 2100, オオシタヤスオ, 43, 990

【解 説】 ㊦ シーケンシャルファイルにデータを追加書き込みします。

- ① シーケンシャルファイルにデータを追加書き込みするときは、ファイルをオープンするときに FOR APPEND を使います。

```
OPEN "meibo.dat" FOR APPEND AS #1
```

↑ ↑ ↑ ↑
ファイルの ファイル名 追加書き込みを指定 オープン番号
オープン

- ② 書き込み、クローズは書き込みを FOR OUTPUT 指定で行ったのと同じです。
- ③ APPEND を使って追加書き込みをするときはファイル名が同じものがあれば、追加で書き込み、すなわち前のデータをこわさないようにオープンします。同じファイル名がないときはその名前でファイルをつくります。つまり OUTPUT を使ったときと同じです。したがって、誤ってファイルをこわさないように OUTPUT より APPEND を使った方が無難です。

【プログラム例】

```
DIM a$(2), x%(2, 2)  ————— 配列の宣言
FOR i = 0 TO 1  ————— データの読み込み
    READ a$(i), x%(i, 1), x%(i, 2)
NEXT i
```



```

OPEN "b:meibo.dat" FOR APPEND AS #3
FOR i = 0 TO 1
    WRITE #3, a$(i), x%(i, 1), x%(i, 2)
NEXT i
CLOSE #3
END
DATA カト`カフヒロシ, 35, 2100, オオシタヤスオ, 43, 990

```

ファイルのオープン
データの追加書き込み
ファイルのクローズ

【結果】 データがディスクに書き込まれます。

MS-DOS の type コマンドでファイルの内容を表示すると次のようになります。

```

"オオヤマタロウ", 38, 1570
"ウエタ`ヨシオ", 41, 1300
"クリタサフ`ロウ", 25, 820
"トクカ`ワヨシオ", 52, 2300
"カワカミタロウ", 21, 420
"アマタ`ヨシミツ", 37, 1210
"カト`カフヒロシ", 35, 2100
"オオシタヤスオ", 43, 990

```

【例題 148】 ◎シーケンシャルファイルの EOF を使った読み出し

【例題 145】と【例題 147】を実行した後のファイル meibo.dat のデータを読み出して表示するプログラムをつくれ。

【解説】 ◎ シーケンシャルファイルのデータを EOF を利用して読み出します。

- ① 一般にシーケンシャルファイルにどれだけデータが入っているかをおぼえていることはありません。そこで、ファイルを読むとき FOR~NEXT 文で読もうと思ってもくり返しの数がわかりません。そこで EOF を利用します。
- ② EOF はファイルにデータがあると 0 (偽)、なくなると -1 (真) を持ちます。これを利用して、ファイルにデータがある間読み出します。

```

DO UNTIL EOF(1)
    INPUT #1, a$, x%, y%
LOOP

```

識別番号
オープン番号 1 のファイルのデータがなくなると EOF(1) は -1 となる

【プログラム例】

```

OPEN "b:meibo.dat" FOR INPUT AS #1
DO UNTIL EOF(1)
    INPUT #1, a$, x%, y%
    PRINT a$; x%; y%
LOOP
CLOSE #1
END

```

ファイルのオープン
データの読み出しと表示
ファイルのクローズ

【結 果】

```

オオヤマトロウ 38 1570
ウエタ`ヨシオ 41 1300
クリタサフ`ロウ 25 820
トクカ`ワヨシオ 52 2300
カワカミタロウ 21 420
アマタ`ヨシミツ 37 1210
カト`カワヒロシ 35 2100
オオシタヤスオ 43 990

```

【例題 149】 シーケンシャルファイルのデータの変更

【例題 148】のファイル meibo.dat の第 3 組目のデータを下のデータで変更するプログラムをつくれ。

データ：タジマコウジ, 55, 1920

【解 説】 ◎ シーケンシャルファイルの 1 部のデータを変更します。

- ① シーケンシャルファイルは前から順に書き、前から順に読み出しますから途中で読み出したり、途中で書いたりすることはできませんでした。Quick BASIC では seek を使ってできるようになっています。この例は II 編 23 章で扱います。本例はまず従来の BASIC と同じ方法によってプログラムします。
- ② 全データを【例題 148】の方法で読み出し配列へ格納します。
- ③ 次に 3 番目を指定して配列データを入れかえます。
- ④ 全データをファイルへ書き込みます。
- ⑤ 読み出すとき EOF() を使うと終りを検出できますが $i=i+1$ によってデータ数を求めておきます。次に書き込むとき、 i 個分をかけばよいことになります。

【プログラム例】

```

DIM aa$(100), xx%(100, 2) ————— 配列の宣言
i = 0 ————— データ組数を数えるカウンタ
OPEN "b:meibo.dat" FOR INPUT AS #2 ————— ファイルのオープン
DO UNTIL EOF(2) ————— データの読み出し配列へ代入
    i = i + 1
    INPUT #2, a$, x%, y%
    aa$(i) = a$
    xx%(i, 1) = x%
    xx%(i, 2) = y%
LOOP
CLOSE #2 ————— ファイルのクローズ
aa$(3) = "タジマコウジ" ————— 新しいデータを代入
xx%(3, 1) = 55
xx%(3, 2) = 1920
OPEN "b:meibo.dat" FOR OUTPUT AS #5 ————— ファイルのオープン
FOR j = 1 TO i ————— データの書き込み
    WRITE #5, aa$(j), xx%(j, 1), xx%(j, 2)
NEXT j
CLOSE #5 ————— ファイルのクローズ
END

```

【結 果】 (ファイルの内容)

```

オオヤマタロウ 38 1570
ウエタ`ヨシオ 41 1300
タシ`マコウシ` 55 1920
トクカ`ワヨシオ 52 2300
カワカミタロウ 21 420
アマタ`ヨシミツ 37 1210
カト`カワヒロシ 35 2100
オオシタヤスオ 43 990

```

【例題 150】 PRINT #を使ってファイルヘデータの書き込み

【例題 145】のデータをファイル meibo2.dat に PRINT #文を使ってかくプログラムをつくれ。

【解 説】 ◎ PRINT #を使ってファイルヘデータをかきます。

- ① WRITE #のかわりに PRINT #を使ってデータをファイルヘかきます。
- ② PRINT #を使うときは文字列は" "でかこんで記入します。"は CHR\$(34)ですから、文字列 a\$ をかくときは次のようになります。

```
PRINT #1,CHR$(34)+a$+CHR$(34)
```

または

```
PRINT #1,CHR$(34);a$;CHR$(34)
```

- ③ 書き込まれたデータの読み出しは普通どおりです。

読み出しプログラムの例とその結果の例は次のとおりです。

【参考プログラム例】

```

OPEN "meibo1.dat" FOR INPUT AS #2
FOR i = 1 TO 6
    INPUT #2, a$, x%, y%
    PRINT a$; x%; y%
NEXT i
CLOSE #2
END

```

ファイルのオープン
ファイルのデータの読み出し
と表示

ファイルのクローズ

【結 果】

```

オオヤマタロウ 38 1570
ウエタ`ヨシオ 41 1300
クリタサフ`ロウ 25 820
トクカ`ワヨシオ 52 2300
カワカミタロウ 21 420
アマタ`ヨシミツ 37 1210

```

【プログラム例】

```

DIM x%(6, 2) ————— 配列の宣言
FOR i = 0 TO 5 ————— 配列ヘデータを読み込む
    READ a$(i), x%(i, 1), x%(i, 2)
NEXT i
k$ = CHR$(34) ————— "を K$に代入
OPEN "b:meibo2.dat" FOR OUTPUT AS #1 ————— ファイルのオープン

```

```

FOR i = 0 TO 5
    PRINT #1, k$ + a$(i) + k$; x%(i, 1); x%(i, 2)
NEXT i
CLOSE #1
END
DATA オオヤマタロウ, 38, 1570, ウエダ ヨシオ, 41, 1300
DATA クリタサブロウ, 25, 820, トカガ ヲヨシオ, 52, 2300
DATA カワカミタロウ, 21, 420, アマダ ヨシミツ, 37, 1210

```

PRINT#を使ってファイルヘデータを書き込む

ファイルのクローズ

【結 果】 データをファイルへ書き込みます。

〔演習問題〕

(152) 下のデータをシーケンシャルファイル `yakyu.dat` にかくプログラムをつくれ。

名 前	ホームラン	打 率
高田 四郎	15	.325
鈴木 太郎	25	.298
戸田 勝美	22	.268
徳田 義男	12	.352

(153) (152) のデータを読み出して表示するプログラムをつくれ。

(154) (152) のデータに追加して次のデータをかくプログラムをつくれ。

名 前	ホームラン	打 率
佐々木歳男	35	.275
大木 伸男	31	.333

(155) (154) のデータを EOF を使って読み出して表示するプログラムをつくれ。

(156) (154) の 2 番目のデータを高田広, 26, .299 で変更するプログラムをつくれ。

(157) 結果のデータ文のデータをシーケンシャルファイル `seisu.dat` にかくプログラムをつくれ。ただし、文字列データとしてかくものとする。

(158) (157) のファイルを使って、全体の和と平均を求めるプログラムをつくれ。

(159) (157) のファイルを使って、整数 `x%` を入力し、`x%` がいくつあるかを表示するプログラムをつくれ。

(160) (157) のファイルを使って、整数 `xx%` を入力し、`xx%` より大きいデータを表示するプログラムをつくれ。

〔 解 答 〕

(152) 【プログラム例】

```

DIM a$(4), x%(4), y!(4) —— 配列の宣言
FOR i = 0 TO 3 —— データの読み込み
    READ a$(i), x%(i), y!(i)
NEXT i
OPEN "b:yakyu.dat" FOR OUTPUT AS #1 —— ファイルのオープン
FOR i = 0 TO 3 —— データの書き込み
    WRITE #1, a$(i), x%(i), y!(i)
NEXT i
CLOSE #1 —— ファイルのクローズ
END
DATA タカダ シロウ, 15, .325, スズキ タロウ, 25, .298
DATA トダ カツミ, 22, .268, トクダ ヨシオ, 12, .352

```

【結 果】 データがファイルへ書き込まれます。

(153) に読み出し方と結果があります。

(153) 【プログラム例】

```

OPEN "b:yakyu.dat" FOR INPUT AS #2 —— ファイルのオープン
FOR i = 1 TO 4 —— データの読み出しと表示
    INPUT #2, a$, x%, y!
    PRINT a$; x%; y!
NEXT i
CLOSE #2 —— ファイルのクローズ
END

```

【結 果】

```

タカダ シロウ 15 .325
スズキ タロウ 25 .298
トダ カツミ 22 .268
トクダ ヨシオ 12 .352

```

(154) 【プログラム例】

```

DIM a$(2), x%(2), y!(2) —— 配列の宣言
FOR i = 0 TO 1 —— データの読み込み
    READ a$(i), x%(i), y!(i)
NEXT i
OPEN "b:yakyu.dat" FOR APPEND AS #3 —— ファイルのオープン
FOR i = 0 TO 1 —— データの追加書き込み
    WRITE #3, a$(i), x%(i), y!(i)
NEXT i
CLOSE #3 —— ファイルのクローズ
END
DATA ササキ トシオ, 35, .275, オオキ ノブオ, 31, .333

```

【結 果】 データが追加書き込みされます。

(155) に読み出し方と結果があります。

(155) 【プログラム例】

```

OPEN "b:yakyu.dat" FOR INPUT AS #1 —— ファイルのオープン
DO UNTIL EOF(1) —— データの読み出しと表示
    INPUT #1, a$, x%, y!
    PRINT a$; x%; y!
LOOP
CLOSE #1 —— ファイルのクローズ
END

```

【結 果】

```

タカタ`シロウ 15 .325
スス`キタロウ 25 .298
トタ`カツミ 22 .268
トクタ`ヨシオ 12 .352
ササキトシオ 35 .275
オオキノフ`オ 31 .333

```

(156) 【プログラム例】

```

DIM aa$(100), xx%(100), yy!(100) —— 配列の宣言
i = 0 —— データ組数を求めるカウンタ
OPEN "b:yakyu.dat" FOR INPUT AS #2 —— ファイルのオープン
DO UNTIL EOF(2) —— データを読み出して配列へ代入
    i = i + 1
    INPUT #2, a$, x%, y!
    aa$(i) = a$
    xx%(i) = x%
    yy!(i) = y!
LOOP
CLOSE #2 —— ファイルのクローズ
aa$(2) = "タカタ`ヒロシ" —— 新しいデータを代入
xx%(2) = 26
yy!(2) = .299
OPEN "b:yakyu.dat" FOR OUTPUT AS #1 —— ファイルのオープン
FOR j = 1 TO i —— データの書き込み
    WRITE #1, aa$(j), xx%(j), yy!(j)
NEXT j
CLOSE #1 —— ファイルのクローズ
END

```

【結 果】 (ファイルの内容)

```

タカタ`シロウ 15 .325
タカタ`ヒロシ 26 .299
トタ`カツミ 22 .268
トクタ`ヨシオ 12 .352
ササキトシオ 35 .275
オオキノフ`オ 31 .333

```

(157) 【プログラム例】

```

DIM x$(50) ——— データを入れる配列の宣言
FOR i = 0 TO 49 ——— データを配列 x$ に代入
    READ x$(i)
NEXT i
OPEN "b:seisu.dat" FOR OUTPUT AS #1 ——— ファイルのオープン
FOR i = 0 TO 49 ——— データをファイルにかく
    WRITE #1, x$(i)
NEXT i
CLOSE #1 ——— ファイルのクローズ
END
DATA 38, 25, 62, 47, 15, 35, 24, 62, 95, 10
DATA 85, 21, 48, 32, 61, 12, 40, 88, 24, 12
DATA 15, 23, 45, 71, 55, 69, 21, 11, 45, 12
DATA 21, 42, 12, 44, 15, 35, 40, 52, 11, 14
DATA 86, 41, 25, 45, 65, 77, 48, 53, 51, 55

```

【結 果】

```

A>type seisu.dat      MS-DOS の TYPE 命令でファイル seisu.dat の内容を表示
"38"
"25"
"62"
"47"
"15"
"35"
"24"
"62"
"95"
"10"
以下省略

```

(158) 【プログラム例】

```

OPEN "b:seisu.dat" FOR INPUT AS #1 ——— ファイルのオープン
tt% = 0 ——— 和を求める変数 tt%
FOR i = 0 TO 49 ——— ファイルのデータを読み数値化して和を求める
    INPUT #1, x$
    tt% = tt% + VAL(x$)
NEXT i
CLOSE #1 ——— ファイルのクローズ
hi = tt% / 50 ——— 平均を hi に求める
PRINT "ワ= "; tt% ——— 結果の表示
PRINT "ハイキン= "; hi
END

```

【結 果】

```

ワ= 2035
ハイキン= 40.7

```


(159) 【プログラム例】

```

OPEN "b:seisu.dat" FOR INPUT AS #1———ファイルのオープン
tt% = 0———カウンタを 0 とする
INPUT x%———x%の入力
FOR i = 0 TO 49———x%と同じ値のとき tt%に 1 を加える
    INPUT #1, x$
    IF x% = VAL(x$) THEN tt% = tt% + 1
NEXT i
CLOSE #1
PRINT x%; tt%; "コ"———結果の表示
END

```

【結 果】

```

? 12          12 を入力の例
12  4 コ
? 11          11 を入力の例
11  2 コ
? 21          21 を入力の例
21  3 コ

```

(160) 【プログラム例】

```

OPEN "b:seisu.dat" FOR INPUT AS #1———ファイルのオープン
INPUT x%———x%の入力
FOR i = 0 TO 49———x%をこえるデータを表示
    INPUT #1, x$
    IF x% < VAL(x$) THEN PRINT x$; " ";
NEXT i
CLOSE #1
PRINT
END

```

【結 果】

```

? 50          50 を入力の例
62 62 95 85 61 88 71 55 69 52 86 65 77 53 51 55
? 80          80 を入力の例
95 85 88 86

```

21 章 ランダムファイル (1)

OPEN~FOR RANDOM, FIELD, LSET, RSET, MKI\$, MKS\$, MKL\$, MKD\$,
CVI, CVS, CVL, CVD, PUT#, GET#, CLOSE#

【例題 151】 ランダムファイル

【例題 145】のデータをランダムファイルにかくプログラムをつくれ。

【解 説】 ㊦ ランダムファイルにデータを書き込みます。

- ① ランダムファイルは書き込み、読み出し、追加の区分はなくすべて FOR RANDOM でオープンします。

```

OPEN  "meibo3.dat" FOR RANDOM AS #1
  ↑      ↑          ↑          ↑
  ファイルの ファイル名 ランダムファイル オープン番号
  オープン              の指定

```

- ② ランダムファイルへの書き込みは Field 文を使ってフィールドを指定しバッファにデータをかいてから、ディスクへ書き込みます。1 バッファは 256 バイトなので、複数組のデータを 1 バッファにつめてかくようにします。この方法はブロッキングによる書き込みといいます。

```
FOR i=0 TO 5
```

```
FIELD #1,14*i AS dd$,10 AS aa$,2 as bb$,2 as cc$
```

フィールドの分割. ③で説明

```
LSET aa$=a$(i)      ——— a$(i)をaa$に代入
```

```
LSET bb$=MKI$(x%(i,0)) ——— x%(i,0), x%(i,1)をbb$,cc$に代入
```

```
LSET cc$=MKI$(x%(i,1))
```

整数を文字化する。長整数は MKL\$, 単精度実数は MKS\$, 倍精度実数は MKD\$を使う。

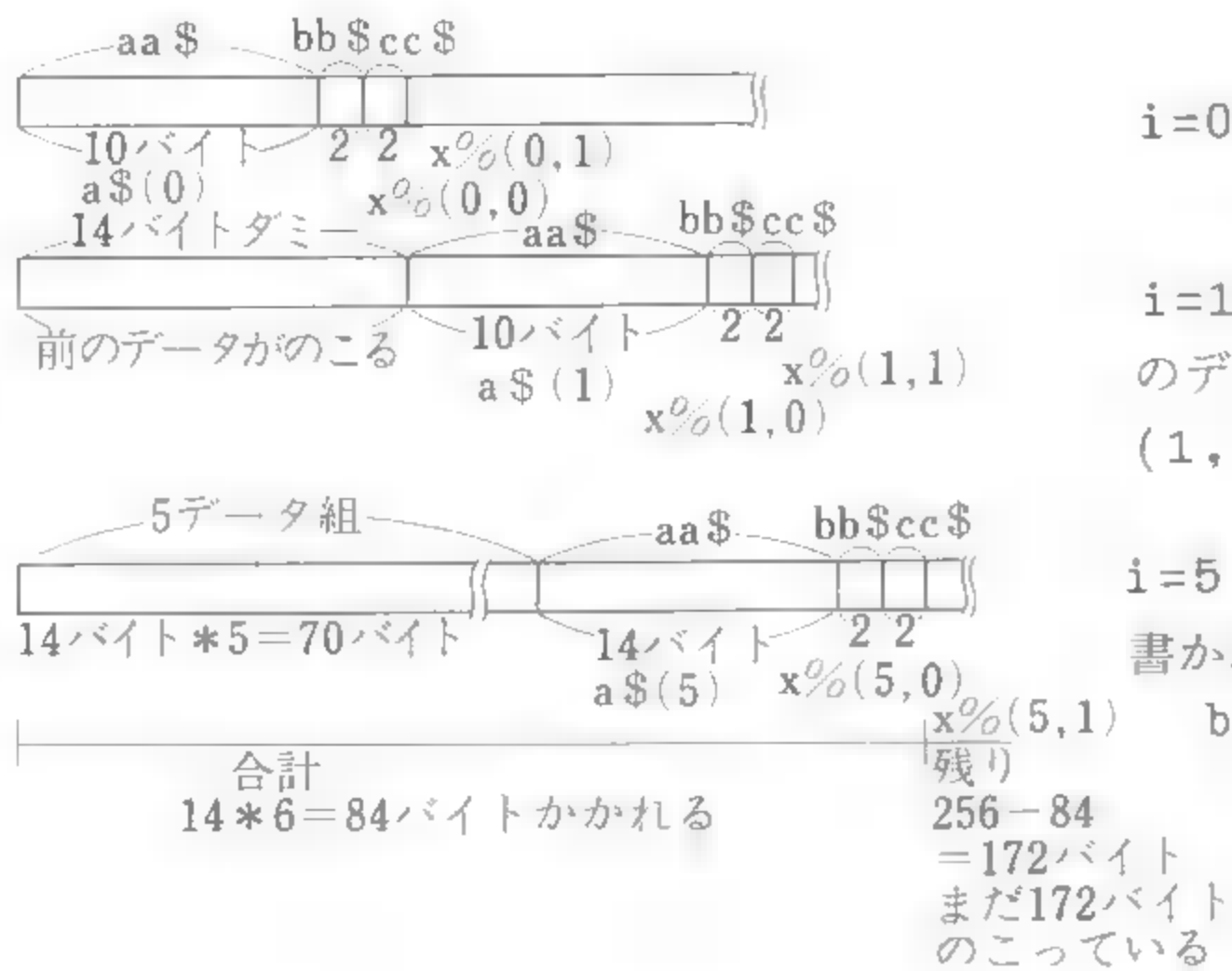
```
NEXT i
```

```
PUT#1,1
```

第1レコードとしてディスクへ書き込み

左づめにセット, 右づめの時は RSET

- ③ フィールドの分割は次のようになります。



i=0 のとき aa\$ は 0 のため aa\$, bb\$, cc\$ を書く

i=1 のとき aa\$ は 14 のため最初の 14 バイトは何も書かれず前のデータがのこる。次の 14 バイトに a\$(1), x%(1,0), x%(1,1) が書き込まれる。

i=5 のとき aa\$ は 70 バイトとなり、最初の 70 バイトには何も書かれず前のデータがのこる。71 バイト目から aa\$(a\$(5)), bb\$(x%(5,0)), cc\$(x%(5,1)) が書かれる。

- ④ ブロッキングを使わないと、FIELD #1, 10 AS aa\$, 2 AS bb\$, 2 AS cc\$ となり、1 組のデータを aa\$, bb\$, cc\$ にセットして PUT #1, 1 から PUT #1, 6 まで 6 レコードに分けて格納します。256 バイトの中にわずか 14 バイトしか使わない無駄となります。

```
FOR i=0 TO 5
    FIELD 10 AS aa$, 2 AS bb$, 2 AS cc$
    LSET aa$=a$(i)
    LSET bb$=MKI$(x%(i,0))
    LSET cc$=MKI$(x%(i,1))
    PUT #1,i+1
```

```
NEXT i
```

- ⑤ なお、2 AS bb\$, bb\$=MKI\$(x%(i,0)) のように整数は MKI\$ を使って文字変数に代入します。その場合の必要バイト数は 2 です。単精度実数の場合は 4 AS bb\$, bb\$=MKS\$(x!(i,0)) のように MKS\$ を使って文字変数に代入します。必要バイト数は 4 です。

長整数の場合は MKL\$ を使います。必要バイト数は 4 です。

倍精度実数の場合は MKD\$ を使います。必要バイト数は 8 です。

【プログラム例】

```
DIM a$(18), X%(18, 2)
FOR i = 0 TO 17
    IF i <= 5 THEN
        READ a$(i), X%(i, 0), X%(i, 1)
    ELSE
        a$(i) = " "
        X%(i, 0) = 0
        X%(i, 1) = 0
    END IF
NEXT i
OPEN "b:meibo3.dat" FOR RANDOM AS #1 LEN = 256
```

—— A\$ は名前, x% は年齢と年収の入る配列
データの配列への読み込み

i が 6 以上は a\$ " ",
x% は 0 となる

—— ファイルのオープン

```

FOR i = 0 TO 17
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    LSET aa$ = a$(i)
    LSET bb$ = MKI$(X%(i, 0))
    LSET cc$ = MKI$(X%(i, 1))
NEXT i
PUT #1, 1
CLOSE #1
END
DATA オヤマタロウ, 38, 1570, ウエダ ヨシオ, 41, 1300
DATA クリタサブ ロウ, 25, 820, トクガ ワシオ, 52, 2300
DATA カワカミタロウ, 21, 420, アマダ ヨシミツ, 37, 1210

```

ファイルへ書き込み

ファイルのクローズ

【例題 152】 ランダムファイルの読み出し

【例題 151】のファイルのデータを読み出し表示するプログラムをつくれ。

【解 説】 ① ランダムファイルのデータを読み出します。

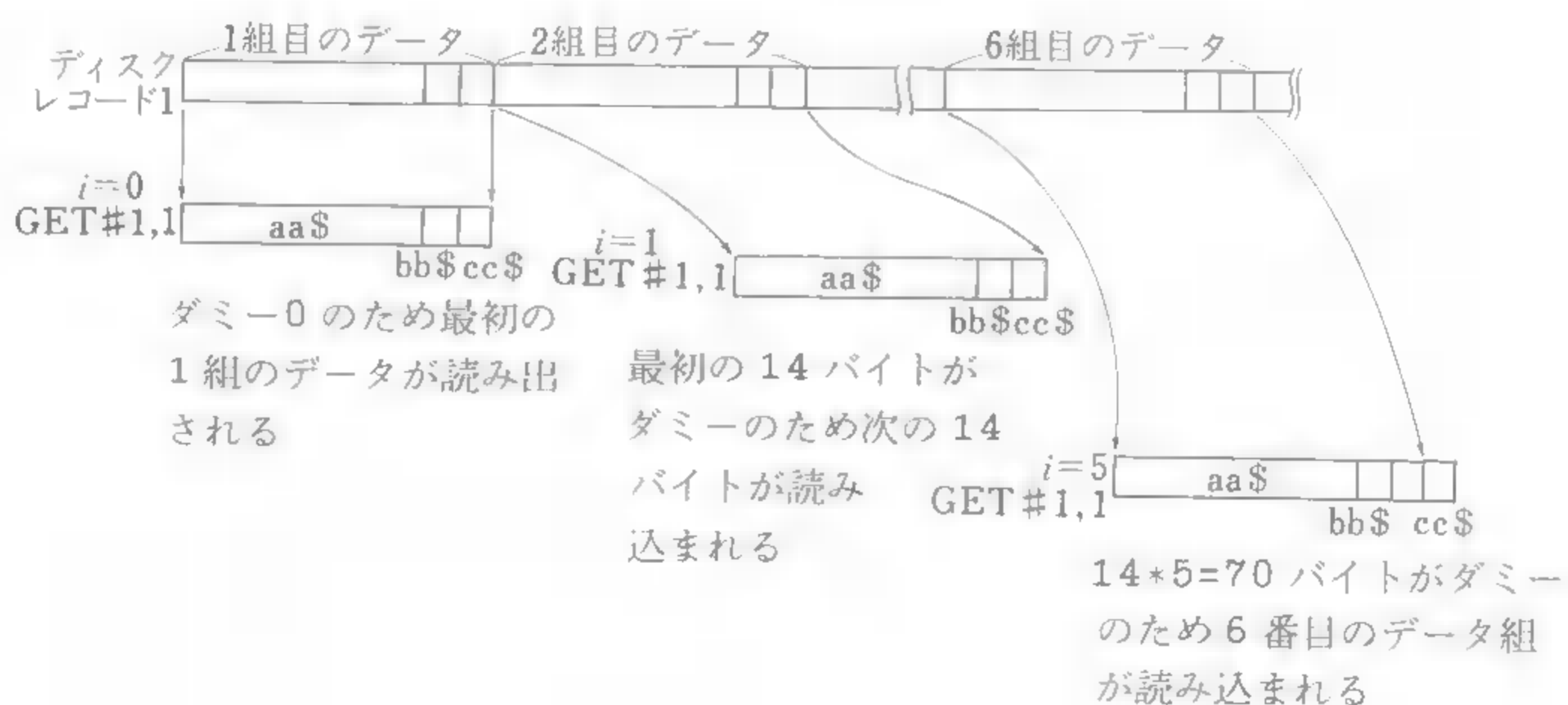
① ランダムファイルの読み出しはまず書き込みと同じ方法でファイルをオープンします。

② 次に書き込みと同じ方法で FIELD を割りあてます。

③ GET #を使ってディスクからデータを読み出します。

GET #1, 1 によってレコード 1 を読み出します。FIELD が【例題 151】と同じとすると $i=0$ のとき最初の 1 組のデータが aa\$, bb\$, cc\$ に読み出されます。 $i=1$ のときは最初の 14 バイトがダミーで次の 14 バイトすなわち 2 組目のデータが aa\$, bb\$, cc\$ に読み出されます。

④ 整数化は CVI を使います。単精度実数の場合は CVS を使います。長整数の場合は CVL, 倍精度実数の場合は CVD を使います。



【プログラム例】

```

DIM a$(18), x%(18, 2)
OPEN "b:meibo3.dat" FOR RANDOM AS #1 LEN = 256

```

配列 a\$ は名前, x% は年齢と年収が入る

ファイルのオープン

```

FOR i = 0 TO 5
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    GET #1, 1
    a$(i) = aa$
    x%(i, 0) = CVI(bb$)
    x%(i, 1) = CVI(cc$)
NEXT i
CLOSE #1
FOR i = 0 TO 5
    PRINT a$(i); x%(i, 0); x%(i, 1)
NEXT i
END

```

読み出し

ファイルのクローズ

表示

【結 果】

オオヤマタロウ	38	1570
ウエタ`ヨシオ	41	1300
クリタリフ`ロウ	25	820
トクカ`ワヨシオ	52	2300
カワカミタロウ	21	420
アマタ`ヨシミツ	37	1210

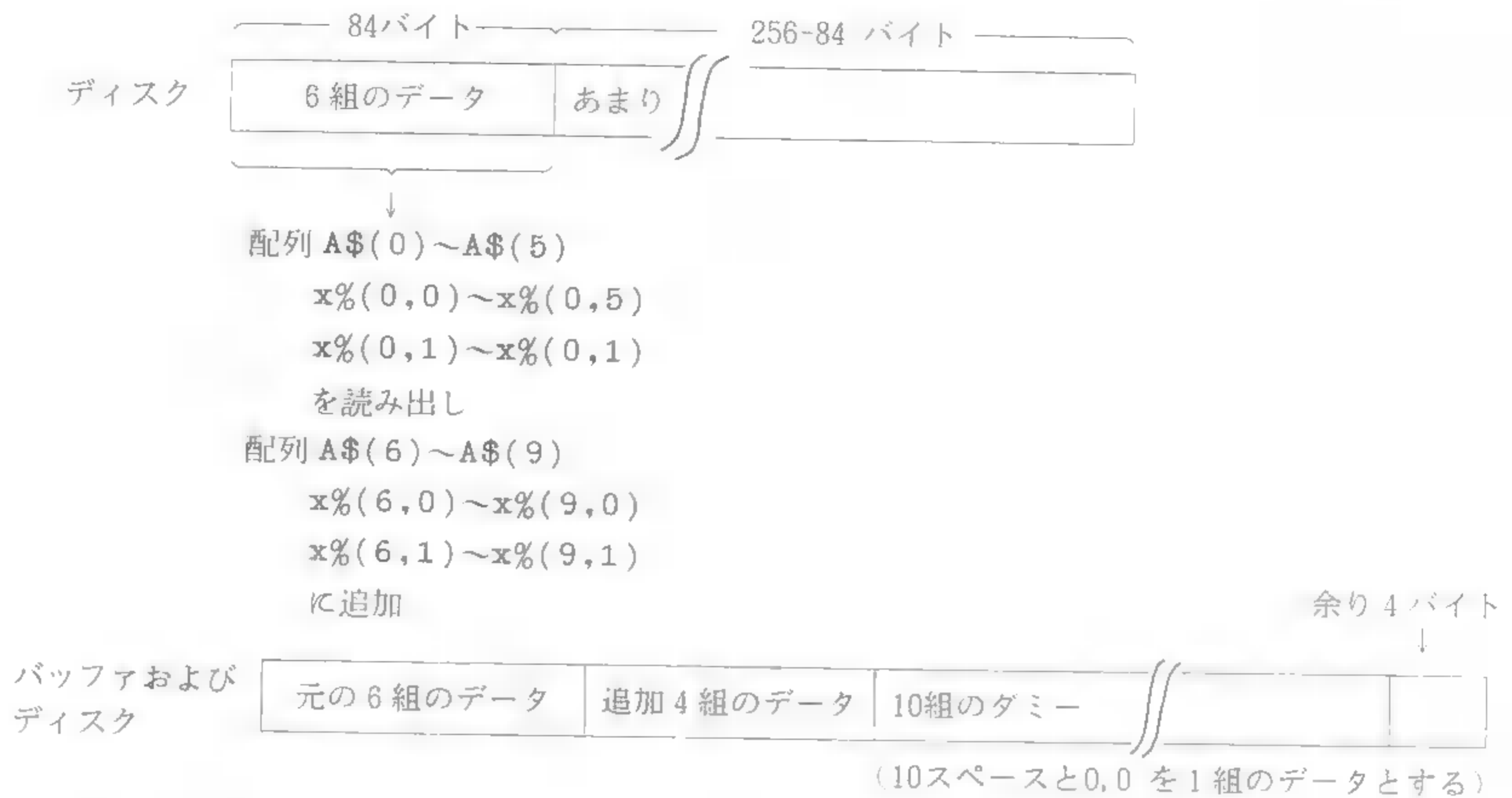
【例題 153】 ランダムファイルヘデータの追加

【例題 151】のファイルに下のデータを追加するプログラムをつくれ。

名 前	年 齢	年 収
川口 等	26	550
太田 浩一	51	1820
岸田 公吉	33	710
和田 吉男	38	830

【解 説】 ㉠ ランダムファイルヘデータを追加

- ① 【例題 151】のランダムファイルは 1 組のデータが文字列 10 文字、整数型、整数型で合計 14 バイトを使用します。したがって、1 バッファ $256 \div 14 = 18$ により 18 データ組が 1 バッファに入ります。
- ② 【例題 151】では 6 データ組のみ入っています。したがって、今回の 4 データ組をその後に追加します。【例題 151】の 6 組と今回の 4 組のデータをかき、後の 8 組はスペースの文字列と数値 0 を記入しておきます。
- ③ 1 度ファイル meibo3.dat のデータを読み出し、配列に格納します。その後に 4 データ組をつけ加えます。
- ④ 合計 10 組をバッファにかきなおします。後の 8 組はスペースと 0 をかき、バッファ内容 18 組のデータをディスクへ格納します。



【プログラム例】

```

DIM a$(18), x%(18, 2)
OPEN "b:meibo3.dat" FOR RANDOM AS #1 LEN = 256
FOR i = 0 TO 17
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    GET #1, 1
    a$(i) = aa$
    IF a$(i) = " " THEN EXIT FOR
    x%(i, 0) = CVI(bb$)
    x%(i, 1) = CVI(cc$)
NEXT i
j = i
FOR i = j TO j + 3
    READ a$(i), x%(i, 0), x%(i, 1)
NEXT i
FOR i = 0 TO j + 3
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    LSET aa$ = a$(i)
    LSET bb$ = MKI$(x%(i, 0))
    LSET cc$ = MKI$(x%(i, 1))
NEXT i
PUT #1, 1
CLOSE #1
END
DATA カゲ ヒト, 26, 550, オタクウイ, 51, 1820
DATA キンダ コウキ, 33, 710, ワダ ヨシオ, 38, 830
    
```

配列 a\$ は名前, x% は年齢と年収が入る

ファイルのオープン

0~17組のデータの配列への読み出し

データ組の追加

追加データ組をディスクセットへ書き込む

ファイルのクローズ

DATA カゲ ヒト, 26, 550, オタクウイ, 51, 1820
DATA キンダ コウキ, 33, 710, ワダ ヨシオ, 38, 830

【例題 154】 ランダムファイルの読み出し

【例題 153】のファイル meibo3.dat を読み出して表示するプログラムをつくれ。

【解 説】 ◎ ランダムファイルの読み出し

- ① 0~18組のデータを読み出し配列へ代入します。その中でデータの入っていないところは名前は

10 文字の空白、年齢と年収は 0 ですから、それをみつけばどこまでデータがあるかわかります。

```
FOR i=0 TO 17
    FIELD 14*i AS aa$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    GET #1, 1
    a$(i)=aa$
    IF a$(i)=" " THEN EXIT FOR
    x%(i,0)=CVI(bb$)
    x%(i,1)=CVI(cc$)
NEXT i
```

—— 第 1 レコードをよみ出す。

—— aa\$に名前が読み出されそれを a\$(i)に代入

—— a\$(i)が空白のときループの外へ

名前が空白でないとき年齢を x%(i, 0),

年収を x%(i, 1)へ代入

データの数は i 個

【プログラム例】

```
DIM a$(18), x%(18, 2)
OPEN "b:meibo3.dat" FOR RANDOM AS #1 LEN = 256
FOR i = 0 TO 17
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    GET #1, 1
    a$(i) = aa$
    IF a$(i) = " " THEN EXIT FOR
    x%(i, 0) = CVI(bb$)
    x%(i, 1) = CVI(cc$)
NEXT i
j = i
CLOSE #1
FOR i = 0 TO j - 1
    PRINT a$(i); x%(i, 0); x%(i, 1)
NEXT i
END
```

—— 名前が入る a\$, 年齢, 年収が x%に入る

—— ファイルのオープン

データ組がなくなるまで名前を a\$(i), 年齢を x%(i, 0), 年収を x%(i, 1)に代入

データ組の表示

【結 果】

オオヤマタロウ	38	1570
ウエタヨシオ	41	1300
クリタサブロー	25	820
トクカワヨシオ	52	2300
カワカミタロウ	21	420
アマタヨシミツ	37	1210
カワグチヒトシ	26	550
オオタコウイチ	51	1820
キシタコウキチ	33	710
ワタヨシオ	38	830

【例題 155】 ランダムファイルの訂正

【例題 153】のファイルの 2 番目を下のデータで書きかえてから全体を表示するプログラムをつくれ。

名前：羽田 修

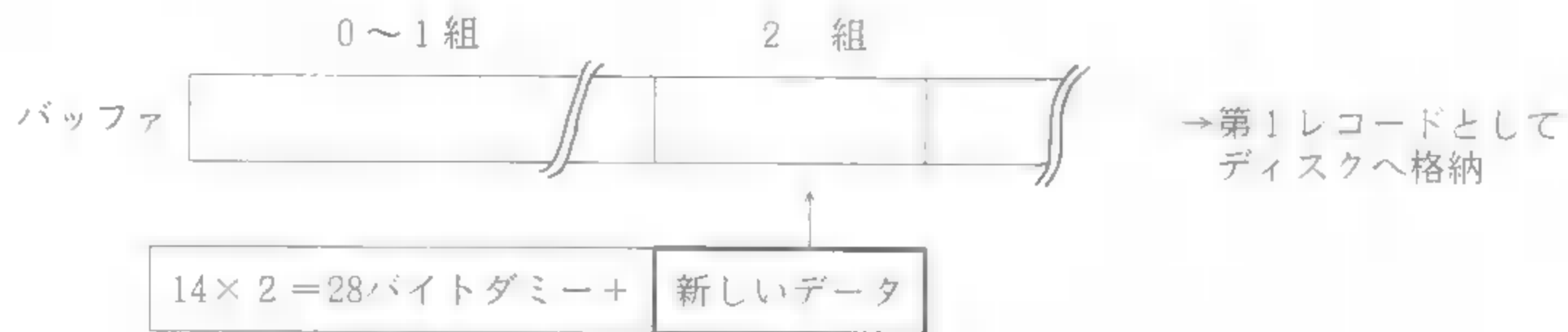
年齢：26 歳

年収：690 万円

【解 説】 ◎ ランダムファイルのデータの変更

- ① ランダムファイルのデータの変更は比較的簡単です。
- ② まず 2 番目のデータ組が何番目のレコードにあるかをさがします。次にそのレコードの何組目かをみつめます。そして、そのレコード番号のみつかった順番のデータ組を前にダミーをつけてバッファにかきこみ、そのレコードをディスクへ格納します。

本例では 1 レコードに 18 データ組が入っていますから 0～17 組が第 1 レコード、18～35 組が第 2 レコード…となります。2 組目は第 1 レコードの 2 番目となります。



- ③ 一般化します。1 レコードは 18 データ組 (0～17 組) が入っていて、第 1 レコードから順にレコード番号があるとします。データ組は 0 番目から始まり、レコードは 1 番目から始まるとします。この時、 x 番目のデータ組は $y+1$ レコードの z 番目となります。

$$x \div 18 = y \text{ あまり } z$$

例 23 番目は $23 \div 18 = 1 \text{ あまり } 5$ したがって 2 レコードの 5 番目

145 番目は $145 \div 18 = 8 \text{ あまり } 1$ したがって 9 レコードの 1 番目

【プログラム例】

```

DIM a$(18), x%(18, 2) ————— 0～17 組のデータ組の入る配列
r = 2 ¥ 18 + 1 ————— 2 番目のレコード組が所属するレコード数
b = 2 MOD 18 ————— 2 番目のレコード組が r レコードの b 番目に属する
OPEN "b:meibo3.dat" FOR RANDOM AS #1 LEN = 256 ———— ファイルのオープン
FOR i = 0 TO 17 ————— r レコードのデータ組を読み出し
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$ ———— 配列に代入
    GET #1, r
    a$(i) = aa$
    x%(i, 0) = CVI(bb$)
    x%(i, 1) = CVI(cc$)
NEXT i
a$(b) = "ハネダ オサム" ————— 訂正データを配列に代入
x%(b, 0) = 26
x%(b, 1) = 690
FOR i = 0 TO 17 ————— r レコードに訂正データを含む
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$ ———— 0～17 組のデータを書き込む
    LSET aa$ = a$(i)
    LSET bb$ = MKI$(x%(i, 0))
    LSET cc$ = MKI$(x%(i, 1))
NEXT i
PUT #1, r
CLOSE #1 ————— ファイルのクローズ
END

```

【結 果】

オオヤマタロウ	38	1570	3 番目を変更
ウエタ`ヨシオ	41	1300	
ハネタ`オサム	26	690	
トクカ`ワヨシオ	52	2300	
カワカミタロウ	21	420	
アマタ`ヨシミツ	37	1210	
カワク`チヒロシ	26	550	
オオタコウイチ	51	1820	
キシタ`コウキチ	33	710	
ワタ`ヨシオ	38	830	

【例題 156】 データの削除

【例題 155】のデータを使い、太田浩一のデータを削除するプログラムをつくれ。

【解 説】 ㉠ データの削除をします。

- ① データの削除は、データを全くなくして、空になった部分をつめてしまう方法や、削除するデータに削除を示すマークをつける方法などがあります。ここでは比較的簡単な削除マークをつける方法を行います。これは全く削除してしまう方法に比べ、必要によって復活できるメリットもあります。
- ② データを読み出し、削除するデータの名前の先頭に"*"マークをつけます。これによって、このデータ組は削除されることがわかります。
- ③ "*"マークをつけたデータを含めてファイルをかきかえます。

【プログラム例】

```

DIM a$(18), x%(18, 2)      データを読み出してオオタコウイチのとき*を先頭につける
OPEN "b:meibo3.dat" FOR RANDOM AS #1 LEN = 256
FOR i = 0 TO 17
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    GET #1, 1
    a$(i) = aa$
    IF a$(i) = "          " THEN EXIT FOR
    IF a$(i) = "オオタコウイチ" THEN a$(i) = "*オオタコウイチ "
    x%(i, 0) = CVI(bb$)
    x%(i, 1) = CVI(cc$)
NEXT i
j = i
FOR i = 0 TO j - 1          ファイルのかきなおし
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    LSET aa$ = a$(i)
    LSET bb$ = MKI$(x%(i, 0))
    LSET cc$ = MKI$(x%(i, 1))
NEXT i
PUT #1, 1
CLOSE #1
END

```

【結 果】 (ファイルの内容)

オオヤマタロウ	38	1570
ウエタ`ヨシオ	41	1300
ハネタ`オサム	26	690

トクカ`ワヨシオ	52	2300	
カワカミタロウ	21	420	
アマタ`ヨシミツ	37	1210	全体を表示するとオオタコウイチは先頭に*がついている
カワク`チヒトシ	26	550	
*オオタコウイチ	51	1820	
キシタ`コウキチ	33	710	
ワタ`ヨシオ	38	830	

【例題 157】 削除したデータを除いてリスト

【例題 156】のデータの一覧表を表示するプログラムをつくれ。

【解 説】 ◎ 削除データを除いて表示します。

- ① 削除データは名前の先頭に"*"がついていますので、名前の先頭をみて"*"であれば表示しません。

【プログラム例】

```

DIM a$(18), x%(18, 2)
OPEN "b:meibo3.dat" FOR RANDOM AS #1 LEN = 256
FOR i = 0 TO 17
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    GET #1, 1
    a$(i) = aa$
    IF a$(i) = " " THEN EXIT FOR
    x%(i, 0) = CVI(bb$)
    x%(i, 1) = CVI(cc$)
NEXT i
j = i
CLOSE #1
FOR i = 0 TO j - 1
    IF LEFT$(a$(i), 1) <> "*" THEN PRINT a$(i); x%(i, 0); x%(i, 1)
NEXT i
END

```

データの読み出し

表示。名前の先頭が
"*"のとき非表示

【結 果】

オオヤマタロウ	38	1570
ウエタ`ヨシオ	41	1300
ハネタ`オサム	26	690
トクカ`ワヨシオ	52	2300
カワカミタロウ	21	420
アマタ`ヨシミツ	37	1210
カワク`チヒトシ	26	550
キシタ`コウキチ	33	710
ワタ`ヨシオ	38	830

【例題 158】 削除データの表示

【例題 156】で削除したデータのリストを表示するプログラムをつくれ。

【解 説】 ◎ 削除データを表示します。

- ① 【例題 157】の逆で、名前の先頭に"*"がついているものを表示します。

【プログラム例】

```

DIM a$(18), x%(18, 2)
OPEN "b:meibo3.dat" FOR RANDOM AS #1 LEN = 256
FOR i = 0 TO 17
    FIELD #1, 14 * i AS dd$, 10 AS aa$, 2 AS bb$, 2 AS cc$
    GET #1, 1
    a$(i) = aa$
    IF a$(i) = "          " THEN EXIT FOR
    x%(i, 0) = CVI(bb$)
    x%(i, 1) = CVI(cc$)
NEXT i
j = i
CLOSE #1
FOR i = 0 TO j - 1
    IF LEFT$(a$(i), 1) = "*" THEN PRINT a$(i); x%(i, 0); x%(i, 1)
NEXT i
END

```

データのよみ出し

*のついているもののみ表示

【結 果】

*オオタコウイチ 51 1820

〔演習問題〕

- (161) 次のデータをランダムファイル meibo4.dat にかくプログラムをつくれ。ただし、名前は 16 文字、出身地と所属は 10 文字分とするものとし、1 レコードに 4 組書き込むものとする。

名 前	年 齢	出 身 地	所 属	年 収 (万円)
遠藤 一郎	33	北 海 道	営業部	465
池田 宏一	36	鹿 児 島 県	総務部	430
中田 浩二	29	大 阪 府	業務部	428
海田 泰二	43	山 口 県	貿易部	580
近藤 三郎	25	青 森 県	業務部	398
小田 憲男	40	富 山 県	経理部	411
山川 一夫	38	東 京 都	営業部	490
河田 安次	30	千 葉 県	営業部	450

- (162) (161) のデータを読み出し表示するプログラムをつくれ。
- (163) (161) のデータを読み出し、40 歳代の人を表示するプログラムをつくれ。
- (164) (161) のデータに次のデータを追加してかくプログラムをつくれ。

名 前	年 齢	出 身 地	所 属	年 収 (万円)
江 田 五 郎	53	東 京 都	営 業 部	1250
小 山 吾 郎	46	愛 知 県	企 画 部	850
井 沢 三 郎	50	高 知 県	営 業 部	1000
高 田 広	53	福 岡 県	業 務 部	1150
大 崎 猛	45	北 海 道	営 業 部	920

(165) (164) のデータを表示するプログラムをつくれ.

(166) (164) のデータのうち高田広の出身地を福岡県から鹿児島県に変更するプログラムをつくれ.

(167) (164) のデータのうち池田宏一のデータを削除するプログラムをつくれ.

〔 解 答 〕

(161) 【プログラム例】

```

DIM a$(16, 3), X%(16, 2) ----- データの読み込み
FOR i = 0 TO 7
    READ a$(i, 0), X%(i, 0), a$(i, 1), a$(i, 2), X%(i, 1)
NEXT i
OPEN "b:meibo4.dat" FOR RANDOM AS #1 LEN = 180 ----- データのファイルへの書き込み
FOR i = 1 TO 2
    FOR j = 0 TO 3
        FIELD #1, 40 * j AS ddd$, 16 AS aa$, 2 AS bb$, 10 AS cc$, 10 AS dd$, 2 AS ee$
        k = (i - 1) * 4 + j
        LSET aa$ = a$(k, 0)
        LSET bb$ = MKI$(X%(k, 0))
        LSET cc$ = a$(k, 1)
        LSET dd$ = a$(k, 2)
        LSET ee$ = MKI$(X%(k, 1))
    NEXT j
    PUT #1, i
NEXT i
CLOSE #1
END
DATA エントウイチロウ, 33, ホッカイトー, イイキョウブ, 465
DATA イケダコウイチ, 36, カゴシマケン, ソウムブ, 430
DATA ナカダコウジ, 29, オオサカフ, キョウムブ, 428
DATA カイダタイジ, 43, ヤマダチケン, ホウエキブ, 580
DATA コントウサブロウ, 25, アオモリケン, キョウムブ, 398
DATA オダノリオ, 40, トヤマケン, ケイリブ, 411
DATA ヤマカワカズオ, 38, トウキョウト, イイキョウブ, 490
DATA カワダヤスジ, 30, チバケン, イイキョウブ, 450

```

(162) 【プログラム例】

```

DIM a$(16, 3), x%(16, 2)
OPEN "b:meibo4.dat" FOR RANDOM AS #1 LEN = 180
FOR i = 1 TO 2
    FOR j = 0 TO 3
        FIELD #1, 40 * j AS ddd$, 16 AS aa$, 2 AS bb$, 10 AS cc$, 10 AS dd$, 2 AS ee$
        GET #1, i
        k = (i - 1) * 4 + j
        a$(k, 0) = aa$
        x%(k, 0) = CVI(bb$)
        a$(k, 1) = cc$
        a$(k, 2) = dd$
        x%(k, 1) = CVI(ee$)
    NEXT j
NEXT i
CLOSE #1
FOR i = 0 TO 7
    PRINT a$(i, 0); x%(i, 0), a$(i, 1); a$(i, 2); x%(i, 1)
NEXT i
END

```

ファイルからデータの読み出し

表示

【結 果】

エント`ウイチロウ	33	ホツカイト`ー	エイキ`ヨウフ`	465
イケタ`コウイチ	36	カコ`シマケン	ソウムフ`	430
ナカタ`コウシ`	29	オウサカフ	キ`ヨウムフ`	428
カイタ`タイシ`	43	ヤマク`チケン	ホ`ウエキフ`	580
コント`ウサフ`ロウ	25	アオモリケン	キ`ヨウムフ`	398
オタ`ノリオ	40	トヤマケン	ケイリフ`	411
ヤマカワカス`オ	38	トウキョウト	エイキ`ヨウフ`	490
カワタ`ヤスシ`	30	チハ`ケン	エイキ`ヨウフ`	450

(163) 【プログラム例】

```

DIM a$(16, 3), x%(16, 2)
OPEN "b:meibo4.dat" FOR RANDOM AS #1 LEN = 180
FOR i = 1 TO 2
    FOR j = 0 TO 3
        FIELD #1, 40 * j AS ddd$, 16 AS aa$, 2 AS bb$, 10 AS cc$, 10 AS dd$, 2 AS ee$
        GET #1, i
        k = (i - 1) * 4 + j
        a$(k, 0) = aa$
        x%(k, 0) = CVI(bb$)
        a$(k, 1) = cc$
        a$(k, 2) = dd$
        x%(k, 1) = CVI(ee$)
    NEXT j
NEXT i
CLOSE #1
FOR i = 0 TO 7
    IF x%(i, 0) < 50 AND x%(i, 0) >= 40 THEN
        PRINT a$(i, 0); x%(i, 0), a$(i, 1); a$(i, 2); x%(i, 1)
    END IF
NEXT i
END

```

データのファイルからの読み出し

40 歳代の人を表示

【結 果】

カイト ^ゝ タイシ ^ゝ	43	ヤマク ^ゝ チケン	ホ ^ゝ ウエキフ ^ゝ	580
オタ ^ゝ ノリオ	40	トヤマケン	ケイリフ ^ゝ	411

(164) 【プログラム例】

```

DIM a$(16, 3), X%(16, 2)
FOR i = 0 TO 4
    READ a$(i, 0), X%(i, 0), a$(i, 1), a$(i, 2), X%(i, 1)
NEXT i
OPEN "b:meibo4.dat" FOR RANDOM AS #1 LEN = 180
FOR i = 3 TO 4
    FOR j = 0 TO 3
        FIELD #1, 40 * j AS ddd$, 16 AS aa$, 2 AS bb$, 10 AS cc$, 10 AS dd$, 2 AS ee$
        k = (i - 3) * 4 + j
        LSET aa$ = a$(k, 0)
        LSET bb$ = MKI$(X%(k, 0))
        LSET cc$ = a$(k, 1)
        LSET dd$ = a$(k, 2)
        LSET ee$ = MKI$(X%(k, 1))
    NEXT j
    PUT #1, i
NEXT i
CLOSE #1
END
DATA エダゴゝロウ, 53, トウキョウト, エイキゝョウフゝ, 1250
DATA コヤマゴゝロウ, 46, アイチケン, キカフゝ, 850
DATA イザワサフゝロウ, 50, コウチケン, エイキゝョウフゝ, 1000
DATA タカダゝヒロシ, 53, フクオカケン, エイキゝョウフゝ, 1150
DATA オオサキツヨシ, 45, ホッカイトゝ, エイキゝョウフゝ, 920

```

データの配列への読み出し

データのファイルへの書き込み

(165) 【プログラム例】

```

DIM a$(16, 3), x%(16, 2)
OPEN "b:meibo4.dat" FOR RANDOM AS #1 LEN = 180
FOR i = 1 TO 4
    FOR j = 0 TO 3
        FIELD #1, 40 * j AS ddd$, 16 AS aa$, 2 AS bb$, 10 AS cc$, 10 AS dd$, 2 AS ee$
        GET #1, i
        k = (i - 1) * 4 + j
        a$(k, 0) = aa$
        x%(k, 0) = CVI(bb$)
        a$(k, 1) = cc$
        a$(k, 2) = dd$
        x%(k, 1) = CVI(ee$)
    NEXT j
NEXT i
CLOSE #1
FOR i = 0 TO 15
    PRINT a$(i, 0); x%(i, 0), a$(i, 1); a$(i, 2); x%(i, 1)
NEXT i
END

```

ファイルのオープン

データの読み出し

ファイルのクローズ

表示

【結 果】

エント`ウイチロウ	33	ホツカイト`ー	エイキ`ヨウフ`	465
イケタ`コウイチ	36	カコ`シマケン	ソウムフ`	430
ナカタ`コウシ`	29	オウサカフ	キ`ヨウムフ`	428
カイト`タイシ`	43	ヤマク`チケン	ホ`ウエキフ`	580
コント`ウサフ`ロウ	25	アオモリケン	キ`ヨウムフ`	398
オタ`ノリオ	40	トヤマケン	ケイリフ`	411
ヤマカワカス`オ	38	トウキヨウト	エイキ`ヨウフ`	490
カワタ`ヤスシ`	30	チハ`ケン	エイキ`ヨウフ`	450
エタ`コ`ロウ	53	トウキヨウト	エイキ`ヨウフ`	1250
コヤマコ`ロウ	46	アイチケン	キカクフ`	850
イサ`ワサフ`ロウ	50	コウチケン	エイキ`ヨウフ`	1000
タカタ`ヒロシ	53	フクオカケン	エイキ`ヨウフ`	1150
オオサキツヨシ	45	ホツカイト`ー	エイキ`ヨウフ`	920
	0			0
	0			0
	0			0

(166) 【プログラム例】

```

DIM a$(16, 3), x%(16, 2)
OPEN "b:meibo4.dat" FOR RANDOM AS #1 LEN = 180
FOR i = 1 TO 4
  FOR j = 0 TO 3
    FIELD #1, 40 * j AS ddd$, 16 AS aa$, 2 AS bb$, 10 AS cc$, 10 AS dd$, 2 AS ee$
    GET #1, i
    k = (i - 1) * 4 + j
    a$(k, 0) = aa$
    x%(k, 0) = CVI(bb$)
    a$(k, 1) = cc$
    a$(k, 2) = dd$
    x%(k, 1) = CVI(ee$)
    IF a$(k, 0) = "タカタ`ヒロシ" THEN a$(k, 1) = "カゴ`シマケン"
  NEXT j
NEXT i
FOR i = 1 TO 4
  FOR j = 0 TO 3
    FIELD #1, 40 * j AS ddd$, 16 AS aa$, 2 AS bb$, 10 AS cc$, 10 AS dd$, 2 AS ee$
    k = (i - 1) * 4 + j
    LSET aa$ = a$(k, 0)
    LSET bb$ = MKI$(x%(k, 0))
    LSET cc$ = a$(k, 1)
    LSET dd$ = a$(k, 2)
    LSET ee$ = MKI$(x%(k, 1))
  NEXT j
  PUT #1, i
NEXT i
CLOSE #1
END

```

データをファイルから読み出し

名前が"タカタヒロシ"のとき出身地を
"カゴシマケン"に変更して配列へ代入

配列データをファイルへかきなおす

【結 果】 (ファイルの内容)

エント`ウイチロウ	33	ホツカイト`ー	エイキ`ヨウフ`	465	
イケタ`コウイチ	36	カコ`シマケン	ソウムフ`	430	
ナカタ`コウシ`	29	オウサカフ	キ`ヨウムフ`	428	
カイタ`タイシ`	43	ヤマク`チケン	ホ`ウエキフ`	580	
コント`ウサフ`ロウ	25	アオモリケン	キ`ヨウムフ`	398	
オタ`ノリオ	40	トヤマケン	ケイリフ`	411	
ヤマカワカス`オ	38	トウキョウト	エイキ`ヨウフ`	490	
カワタ`ヤスシ`	30	チハ`ケン	エイキ`ヨウフ`	450	
エタ`コ`ロウ	53	トウキョウト	エイキ`ヨウフ`	1250	
コヤマコ`ロウ	46	アイチケン	キカクフ`	850	
イサ`ワサフ`ロウ	50	コウチケン	エイキ`ヨウフ`	1000	出身地が福岡県から鹿児島県
タカタ`ヒロシ	53	カコ`シマケン	エイキ`ヨウフ`	1150	に変更されている
オオサキツヨシ	45	ホツカイト`ー	エイキ`ヨウフ`	920	
	0			0	
	0			0	
	0			0	

(167)【プログラム例】

```

DIM a$(16, 3), x%(16, 2)
OPEN "b:meibo4.dat" FOR RANDOM AS #1 LEN = 180
FOR i = 1 TO 4
  FOR j = 0 TO 3
    FIELD #1, 40 * j AS ddd$, 16 AS aa$, 2 AS bb$, 10 AS cc$, 10 AS dd$, 2 AS ee$
    GET #1, i
    k = (i - 1) * 4 + j
    a$(k, 0) = aa$
    x%(k, 0) = CVI(bb$)
    a$(k, 1) = cc$
    a$(k, 2) = dd$
    x%(k, 1) = CVI(ee$)
    IF a$(k, 0) = "イケダ`コウイチ" THEN a$(k, 0) = "*イケダ`コウイチ"
  NEXT j
NEXT i
FOR i = 1 TO 4
  FOR j = 0 TO 3
    FIELD #1, 40 * j AS ddd$, 16 AS aa$, 2 AS bb$, 10 AS cc$, 10 AS dd$, 2 AS ee$
    k = (i - 1) * 4 + j
    LSET aa$ = a$(k, 0)
    LSET bb$ = MKIS(x%(k, 0))
    LSET cc$ = a$(k, 1)
    LSET dd$ = a$(k, 2)
    LSET ee$ = MKIS(x%(k, 1))
  NEXT j
  PUT #1, i
NEXT i
CLOSE #1
END

```

ファイルからデータを読み出し、名前が
"イケダコウイチ" のとき先頭に*をつけて
配列へ代入

配列データをファイルへかきなおす

【結 果】(ファイルの内容)

エント`ウイチロウ	33	ホッカイト`ー	エイキ`ヨウフ`	465	
*イクタ`コウイチ	36	カコ`シマケン	ソウムフ`	430	——削除マークがついている
ナカタ`コウシ`	29	オウサカフ	キ`ヨウムフ`	428	
カイト`タイシ`	43	ヤマク`チケン	ホ`ウエキフ`	580	
コント`ウサフ`ロウ	25	アオモリケン	キ`ヨウムフ`	398	
オタ`ノリオ	40	トヤマケン	ケイリフ`	411	
ヤマカワカス`オ	38	トウキョウト	エイキ`ヨウフ`	490	
カワタ`ヤスシ`	30	チハ`ケン	エイキ`ヨウフ`	450	
エタ`コ`ロウ	53	トウキョウト	エイキ`ヨウフ`	1250	
コヤマコ`ロウ	46	アイチケン	キカクフ`	850	
イサ`ワサフ`ロウ	50	コウチケン	エイキ`ヨウフ`	1000	
タカタ`ヒロシ	53	カコ`シマケン	エイキ`ヨウフ`	1150	
オオサキツヨシ	45	ホッカイト`ー	エイキ`ヨウフ`	920	
	0			0	
	0			0	
	0			0	

22 章 ランダムファイル (2)

TYPE 型を使用 LOF

【例題 159】 TYPE 型を使ったランダムファイルへの書き込み

下のデータ組をランダムファイル computer.dat にかくプログラムをつくれ.

データ:	型名	社名	価格 (円)
	X6800	Sharp	356000
	FM-Towns	Fujitsu	338000
	PC-9801ES/2	NEC	448000

【解説】 ◎ TYPE 型を使ってランダムファイルヘデータ組を書き込みます。

- ① Quick BASIC ではバッファを Field 文で割りあてる従来の方法の他に TYPE 型を使って、ランダムファイルをつくることができます。この場合は 1 バッファが 256 バイトと固定した考えでなく、TYPE 型の持つバイト数で決められるためむだが少なくてすみます。
- ② TYPE 型でタブ computer を宣言します。文字型 15+10 バイトと長整数 4 バイトの合計 29 バイトが 1 組の必要バイト数です。
- ③ ランダムファイルのオープンは次のようにします。

[illegible]

LEN=LEN(a)によってバッファのバイト数が決められますから、無駄なく書き込みができます。

- ④ 次の命令でデータ組を書き込みます。

PUT #1, i, a

| | | |

書き込み オープン番号 レコード番号 書き込む内容

a.brand, a.company, a.price を1組として一度に書き込む

i が 1, 2, 3 となり 1レコード, 2レコード, 3レコードを順にかく。

- ⑤ ファイルのクローズは **CLOSE** #1 とオープン番号を使って行います。

【プログラム例】

```

TYPE computer
    brand AS STRING * 15
    company AS STRING * 10
    price AS LONG
END TYPE
DIM a AS computer
OPEN "b:computer.dat" FOR RANDOM AS #1 LEN = LEN(a)
FOR i = 1 TO 3
    READ a.brand, a.company, a.price
    PUT #1, i, a
NEXT i
CLOSE #1
END
DATA X6800, Sharp, 356000, FM-Towns, Fujitsu, 338000
DATA PC-9801ES/2, NEC, 448000

```

TYPE 型の宣言
 TYPE 型の変数 a を宣言
 ファイルのオープン
 1 組 3 つのデータ組を 3 組読み込んで
 レコード番号 1, 2, 3 としてファイル
 へ書き込む
 ファイルのクローズ

【例題 160】 ランダムファイルの読み出し

【例題 159】で書き込んだデータを読み出して表示するプログラムをつくれ。

【解 説】 ② ランダムファイルの読み出しをします。

- ① 読み出したデータを格納するタイプ型変数の宣言をします。これは書き込みした時と同じ TYPE 型とします。
- ② ファイルをオープンします。ファイルのオープンは書き込みと全く同じです。
- ③ 次の命令でデータ組を読み出します。

```

GET #1, i, a

```

読み出し オープン レコード番号
 番号 i を 1, 2, 3 として 1 レコード, 2 レコード, 3 レコードを読み出す
 タイプ型変数 3 データを 1 組として読み出す

【プログラム例】

```

TYPE computer
    brand AS STRING * 15
    company AS STRING * 10
    price AS LONG
END TYPE
DIM a AS computer
OPEN "b:computer.dat" FOR RANDOM AS #1 LEN = LEN(a)
FOR i = 1 TO 3
    GET #1, i, a
    PRINT a.brand; a.company; a.price
NEXT i
CLOSE #1
END

```

TYPE 型の宣言
 TYPE 型の変数 a を宣言
 ファイルのオープン
 ファイルからデータを読み出して表示
 ファイルのクローズ

【結 果】

X6800	Sharp	356000
FM-Towns	Fujitsu	338000
PC-9801ES/2	NEC	448000

【例題 161】 ランダムファイルへ追加

【例題 159】でつくったファイル `computer.dat` に下のデータ組を追加して書き込むプログラムをつくれ。

データ:	型名	社名	価格
	PC-286VE	Epson	298000
	J-3100	Toshiba	448000

【解 説】 ㉓ ランダムファイルにデータ組を追加します。

- ① TYPE 型変数を使ってランダムファイルをつくる場合、1 組のデータが 1 レコードとして書き込まれていますから、レコード組の追加は書き込まれている最後のレコードの次に書き込めばよいことになります。
- ② 書き込まれているレコード数は次の式で求められます。

$$\frac{\text{LOF}(1)}{\text{LEN}(a)} \quad \cdots \cdots \text{レコード数となる}$$

\uparrow \uparrow
 全体のバイト数 TYPE 型変数の
 総バイト数

- ③ したがって、追加でかき込むためには上の $\text{LOF}(1)/\text{LEN}(a)$ の値+1 番目のレコードからかきこめばよいことになります。

【プログラム例】

```

TYPE computer ————— TYPE 型の宣言
    brand AS STRING * 15
    company AS STRING * 10
    price AS LONG
END TYPE —————
DIM a AS computer ————— TYPE 型の変数の宣言
OPEN "b:computer.dat" FOR RANDOM AS #1 LEN = LEN(a) ——— ファイルのオープン
n = LOF(1) / LEN(a) ————— レコード数を n に求める
FOR i = n + 1 TO n + 2 ————— n+1 レコード、n+2 レコードに
    READ a.brand, a.company, a.price ——— データを書き込む
    PUT #1, i, a
NEXT i ————— ファイルのクローズ
CLOSE #1
END
DATA PC-286VE, Epson, 298000, J-3100, Toshiba, 448000
  
```

【例題 162】 ファイルデータの読み込み

【例題 161】を行った後のファイル `computer.dat` のデータを読み出して表示するプログラムをつくれ。

【解 説】 ㉟ ファイルデータを読み出します。レコード数の数え方を確認します。

① ファイル `computer.dat` 中のレコード数は不明です。レコード数は次の式で求めます。

$$\text{LOF}(1) / \text{LEN}(a) \quad \dots \text{レコード数}$$

\uparrow \uparrow
 全体バイト数 TYPE 型変数の
 バイト数

② レコード数が n 個とわかれば 1 から n までのくり返しで n レコードを順に読み出し表示します。

【プログラム例】

```

TYPE computer
    brand AS STRING * 15
    company AS STRING * 10
    price AS LONG
END TYPE
DIM a AS computer
OPEN "b:computer.dat" FOR RANDOM AS #1 LEN = LEN(a)
n = LOF(1) / LEN(a)
FOR i = 1 TO n
    GET #1, i, a
    PRINT a.brand; a.company; a.price
NEXT i
CLOSE #1
END
  
```

TYPE computer — TYPE 型の宣言
 DIM a AS computer — TYPE 型変数 a の宣言
 OPEN "b:computer.dat" FOR RANDOM AS #1 LEN = LEN(a) — ファイルのオープン
 n = LOF(1) / LEN(a) — レコード数を n に求める
 FOR i = 1 TO n — 1 から n レコードまでを
 GET #1, i, a 読み出して表示
 PRINT a.brand; a.company; a.price
 NEXT i
 CLOSE #1 — ファイルのクローズ
 END

【結 果】

X6800	Sharp	356000
FM-Towns	Fujitsu	338000
PC-9801ES/2	NEC	448000
PC-286VE	Epson	298000
J-3100	Toshiba	448000

【例題 163】 ランダムファイルの書き換え

ファイル `computer.dat` の第 2 レコードを下のデータ組に入れかえるプログラムをつくれ。

データ: FM77AV40sx, Fujitsu, 267800

【解 説】 ㉟ ランダムファイルの書き換えをします。

① ランダムファイルのレコードの書き換えは、書き換えるレコード番号を指定して、新しいデータ組を書き込めば変更が行われます。


```
PUT #1, 2, a
```

オープン番号 | TYPE 型変数 a (新しいデータ組)
第 2 レコードに書き込む

【プログラム例】

```
TYPE computer
  brand AS STRING * 15
  company AS STRING * 10
  price AS LONG
END TYPE
DIM a AS computer
OPEN "b:computer.dat" FOR RANDOM AS #1 LEN = LEN(a)
READ a.brand, a.company, a.price
PUT #1, 2, a
CLOSE #1
END
DATA FM77AV40sx, Fujitsu, 267800
```

TYPE 型の宣言
TYPE 型変数 a の宣言
ファイルのオープン
データの読み込み
2 レコード目に書き込み
ファイルのクローズ

【結果】 (ファイルの内容)

X6800	Sharp	356000	
FM77AV40sx	Fujitsu	267800	—— 変更データ
PC-9801ES/2	NEC	448000	
PC-286VE	Epson	298000	
J-3100	Toshiba	448000	

【例題 164】 ファイルデータを配列に読み出す

ファイル computer.dat のデータ組を TYPE 型配列変数 b() に読み出して表示するプログラムをつくれ。

【解 説】 ㊦ ファイルのデータ組を配列へ読み出します。

- ① ファイルのデータ組を読み出し、配列へ代入します。
- ② 配列を TYPE 型の配列変数とすれば次のようになります。

```
FOR i=1 TO n
  GET #1, i, a
  b(i).brand=a.brand
  b(i).company=a.company
  b(i).price=a.price
NEXT i
```

レコード数 n 個、i を 1 から n までくり返す
TYPE 型変数 a に i 番目のレコードを読む
a.brand ~ a.price を b(i).brand ~ b(i).price に代入

- ③ TYPE 型配列変数ではなく 2 次元配列に代入する場合は次のように代入できます。

```
b$(i,1)=a.brand    b(i).brand のかわりに b$(i,1)を使う
b$(i,2)=a.company  b(i).company のかわりに b$(i,2)を使う
c(i)=a.price        b(i).price のかわりに c(i)を使う
```

前もって配列 DIM b\$(1 TO n, 1 TO 2) と DIM c(i) をしておく。

【プログラム例】

```

TYPE computer
    brand AS STRING * 15
    company AS STRING * 10
    price AS LONG
END TYPE
DIM a AS computer
OPEN "b:computer.dat" FOR RANDOM AS #1 LEN = LEN(a)
n = LOF(1) / LEN(a)
DIM b(1 TO n) AS computer
FOR i = 1 TO n
    GET #1, i, a
    b(i).brand = a.brand
    b(i).company = a.company
    b(i).price = a.price
NEXT i
CLOSE #1
FOR i = 1 TO n
    PRINT b(i).brand; b(i).company; b(i).price
NEXT i
END

```

TYPE 型の宣言

TYPE 型変数 a の宣言

ファイルのオープン

レコード数を n に求める

TYPE 型配列変数 b() を宣言

ファイルのデータを読み出して配列変数へ代入

ファイルのクローズ

表示

【結 果】

X6800	Sharp	356000
FM77AV40sx	Fujitsu	267800
PC-9801ES/2	NEC	448000
PC-286VE	Epson	298000
J-3100	Toshiba	448000

【例題 165】 ファイルの検索

ファイル computer.dat を使って、整数 x% を入力し、x% 番目のレコードのデータ組を表示するプログラムをつくれ。

【解 説】 ◎ ファイルデータを検索します。

- ① ランダムファイルのデータの検索は、レコード番号を指定して、そのレコードのデータ組を TYPE 型変数に読み出します。

【プログラム例】

```

TYPE computer
    brand AS STRING * 15
    company AS STRING * 10
    price AS LONG
END TYPE
DIM a AS computer
INPUT x%
OPEN "b:computer.dat" FOR RANDOM AS #1 LEN = LEN(a)
GET #1, x%, a
PRINT a.brand; a.company; a.price

```

TYPE 型の宣言

TYPE 型変数 a の宣言

整数 x% の入力

ファイルのオープン

x% 番目のレコードを読み出して表示

〔 解 答 〕

(168) 【プログラム例】

```

TYPE meibo
    namae AS STRING * 10
    nenrei AS INTEGER
    nenshu AS INTEGER
END TYPE
DIM a AS meibo
OPEN "b:meibo5.dat" FOR RANDOM AS #1 LEN = LEN(a)
FOR i = 1 TO 6
    READ a.namae, a.nenrei, a.nenshu
    PUT #1, i, a
NEXT i
CLOSE #1
END
DATA オオヤマタロウ, 38, 1570, ウエダ ヨシオ, 41, 1300
DATA クリタサフ ロウ, 25, 820, トクガ ワヨシオ, 52, 2300
DATA カワカミタロウ, 21, 420, アマダ ヨシミツ, 37, 1210

```

TYPE 型の宣言
 TYPE 型変数 a の宣言
 ファイルのオープン
 データを読んでファイルへ書き込み
 ファイルのクローズ

(169) 【プログラム例】

```

TYPE meibo
    namae AS STRING * 10
    nenrei AS INTEGER
    nenshu AS INTEGER
END TYPE
DIM a AS meibo
OPEN "b:meibo5.dat" FOR RANDOM AS #1 LEN = LEN(a)
FOR i = 1 TO 6
    GET #1, i, a
    PRINT a.namae; a.nenrei; a.nenshu
NEXT i
CLOSE #1
END

```

TYPE 型の宣言
 TYPE 型変数 a の宣言
 ファイルのオープン
 ファイルから 6 組のデータを読み込み
 ファイルのクローズ

【結 果】

オオヤマタロウ	38	1570
ウエダ ヨシオ	41	1300
クリタサフ ロウ	25	820
トクガ ワヨシオ	52	2300
カワカミタロウ	21	420
アマダ ヨシミツ	37	1210

(170) 【プログラム例】

```

TYPE meibo
    namae AS STRING * 10
    nenrei AS INTEGER
    nenshu AS INTEGER
END TYPE
DIM a AS meibo
OPEN "b:meibo5.dat" FOR RANDOM AS #1 LEN = LEN(a)
n = LOF(1) / LEN(a)
FOR i = n + 1 TO n + 3
    READ a.namae, a.nenrei, a.nenshu
    PUT #1, i, a
NEXT i
CLOSE #1
END
DATA シミズ リョウ, 38, 950, チハ カットシ, 25, 880
DATA ハシダ テツオ, 39, 730

```

TYPE meibo — TYPE 型の宣言
 namae AS STRING * 10
 nenrei AS INTEGER
 nenshu AS INTEGER
 END TYPE
 DIM a AS meibo — TYPE 型変数 a の宣言
 OPEN "b:meibo5.dat" FOR RANDOM AS #1 LEN = LEN(a) — ファイルのオープン
 n = LOF(1) / LEN(a) — レコード数を n に求める
 FOR i = n + 1 TO n + 3 — ファイルに 3 組のレコードを追加
 READ a.namae, a.nenrei, a.nenshu
 PUT #1, i, a
 NEXT i
 CLOSE #1 — ファイルのクローズ
 END
 DATA シミズ リョウ, 38, 950, チハ カットシ, 25, 880
 DATA ハシダ テツオ, 39, 730

(171) 【プログラム例】

```

TYPE meibo
    namae AS STRING * 10
    nenrei AS INTEGER
    nenshu AS INTEGER
END TYPE
DIM a AS meibo
OPEN "b:meibo5.dat" FOR RANDOM AS #1 LEN = LEN(a)
n = LOF(1) / LEN(a)
FOR i = 1 TO n
    GET #1, i, a
    PRINT a.namae; a.nenrei; a.nenshu
NEXT i
CLOSE #1
END

```

TYPE meibo — TYPE 型の宣言
 namae AS STRING * 10
 nenrei AS INTEGER
 nenshu AS INTEGER
 END TYPE
 DIM a AS meibo — TYPE 型変数 a の宣言
 OPEN "b:meibo5.dat" FOR RANDOM AS #1 LEN = LEN(a) — ファイルのオープン
 n = LOF(1) / LEN(a) — レコード数を n に求める
 FOR i = 1 TO n — ファイルのデータを読み出して表示
 GET #1, i, a
 PRINT a.namae; a.nenrei; a.nenshu
 NEXT i
 CLOSE #1 — ファイルのクローズ
 END

【結 果】

オオヤマタロウ	38	1570
ウエタ ヨシオ	41	1300
クリタサフ ロウ	25	820
トクカ ワヨシオ	52	2300
カワカミタロウ	21	420
アマタ ヨシミツ	37	1210
シミズ リョウ	38	950
チハ カットシ	25	880

(172) 【プログラム例】

```

TYPE meibo
    namae AS STRING * 10
    nenrei AS INTEGER
    nenshu AS INTEGER
END TYPE
DIM a AS meibo
READ a.namae, a.nenrei, a.nenshu
OPEN "b:meibo5.dat" FOR RANDOM AS #1 LEN = LEN(a)
PUT #1, 3, a
CLOSE #1
END
DATA ソノダ センイチ, 62, 5230

```

TYPE 型の宣言

TYPE 型変数 a の宣言

データ文のデータを読む

ファイルのオープン

第 3 レコード数へ書き込み

ファイルのクローズ

【結 果】 (ファイルの内容)

オオヤマタロウ	38	1570	
ウエタ ヨシオ	41	1300	
ソノダ センイチ	62	5230	変更データ
トクカ ワヨシオ	52	2300	
カワカミタロウ	21	420	
アマタ ヨシミツ	37	1210	
シミス リョウ	38	950	
チハ カツトシ	25	880	
ハシタ テツオ	39	730	

(173) 【プログラム例】

```

TYPE meibo
    namae AS STRING * 10
    nenrei AS INTEGER
    nenshu AS INTEGER
END TYPE
DIM a AS meibo
OPEN "b:meibo5.dat" FOR RANDOM AS #1 LEN = LEN(a)
GET #1, 4, a
PRINT a.namae; a.nenrei; a.nenshu
CLOSE #1
END

```

タイプ型の宣言

配列の宣言

ファイルのオープン

4 番目のデータの読み出し

表示

ファイルのクローズ

【結 果】

トクカ ワヨシオ 52 2300

23 章 シーケンシャルファイルを ランダムファイル化

SEEK

【例題 166】 SEEK を使ってシーケンシャルファイルをランダムファイルのように使う

下のデータをシーケンシャルファイル reidai.dat にかき、次に整数 xx% を 1 つ入力し、xx% 番目のデータを読み出して表示するプログラムをつくれ。

データ： トウキョウ、オオサカ、タカマツ、サッポロ、チバ、センダイ、ヒロシマ、
ナガサキ、トリデ、ミヤザキ

【解 説】 ◎ SEEK を使うとシーケンシャルファイルをランダムファイルのように使えます。

- ① SEEK は指定した位置へファイルポインタをおくことができます。したがって、その位置から読み出したり、その位置へ書き込みをしたりすることができます。
- ② シーケンシャルファイルは数値型、文字型に関係なく順に書き込み、読み出しができます。また、そのとき、文字型の文字の数が一定でなくてもよいという便利さがあります。
- ③ しかし、ファイルポインタを任意の場所においても、特に文字型データで長さがばらばらではどこを指定してよいのかわかりません。したがって、ここでは、10 個の都市名のデータをおのおの 10 文字の長さとして格納することにします。
- ④ まず 10 都市名をシーケンシャルファイル reidai.dat にかきます。次がそのプログラム例です。

【プログラム例】

```

DIM x$(10)
FOR i = 0 TO 9
    READ x$(i)
NEXT i
OPEN "b:reidai.dat" FOR OUTPUT AS #1
FOR i = 0 TO 9
    WRITE #1, x$(i)
NEXT i
CLOSE #1
END
DATA "トウキョウ"    ", "オオサカ    "
DATA "タカマツ"    ", "サッポロ    "
DATA "チバ"        ", "センダイ    "
DATA "ヒロシマ"    ", "ナガサキ    "
DATA "トリデ"      ", "ミヤザキ    "

```


- ⑤ 確認のため順に読み出すと次のようになります。

【プログラム例】 読み出しのプログラム

```
OPEN "b:reidai.dat" FOR INPUT AS #1
FOR i = 0 TO 9
  INPUT #1, x$
  PRINT x$
NEXT i
CLOSE #1
END
```

【結 果】 読み出し結果

トウキョウ
オオサカ
タカマツ
サッポロ
チハ
センタ
ヒロシマ
ナカサキ
トリデ
ミヤサキ

- ⑥ 次に SEEK を使ってファイルポインタを移動させます。

10 個の都市名は次のように格納されています。

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
										10 文字										10 文字														
" トウキョウ□□□□□□"														" オオサカ□□□□□□"														" タカマツ□....."						
14 桁														14 桁																				

- ⑦ したがって $xx\%$ 番目のデータは $(xx\% - 1) * 14 + 1$ バイト目からかかれています。そこで、そこへファイルポインタを移します。ファイルポインタは SEEK を使って自由に動かします。指定位置へファイルポインタを移して、読み出し、書き込みします。

```
SEEK #1, y%
```

| └── バイト番号 ($xx\%$ 番目のときは $(xx\% - 1) * 14 + 1$ が $y\%$ の値となる)
| └── ファイルのオープン番号

- ⑧ ファイルポインタは任意の位置へ動きますからあたかもシーケンシャルファイルがランダムファイルのように扱えます。

【プログラム例】

```
INPUT xx% ----- xx% の入力
OPEN "b:reidai.dat" FOR INPUT AS #1 ----- ファイルのオープン
y% = (xx% - 1) * 14 + 1 ----- xx% 番目のデータの先頭のバイト数
SEEK #1, y% ----- ファイルポインタを y% へ移す
INPUT #1, x$ ----- y% からの 1 データを読み出す
CLOSE #1 ----- ファイルのクローズ
PRINT xx%; "ハソメ="; x$ ----- 表示
END
```

【結 果】

? 3	3 を入力の場合
3 ハンメ=タカマツ	
? 4	4 を入力の場合
4 ハンメ=サツホロ	
? 10	10 を入力の場合
10 ハンメ=ミヤサキ	
? 6	6 を入力の場合
6 ハンメ=センタイ	

【例題 167】 整数ファイルを seek でさがす

次のデータをシーケンシャルファイル reidai1.dat にかき、次に番号を指定して、その番号のデータを読み出して表示するプログラムをつくれ。

データ: 5, 3, 2, 7, 8, 9, 1, 5, 4, 9
 9, 4, 8, 2, 7, 4, 9, 1, 3, 0

【解 説】 ◎ 整数ファイルを seek を使ってさがします。

① 整数データをファイル reidai1.dat にかきます。

【プログラム例】

```

DIM x%(20)
FOR i = 0 TO 19 ————— データの読み込み
  READ x%(i)
NEXT i
OPEN "b:reidai1.dat" FOR OUTPUT AS #1 ————— データをファイルへ書き込み
FOR i = 0 TO 19
  WRITE #1, x%(i)
NEXT i
CLOSE #1
END
DATA 5, 3, 2, 7, 8, 9, 1, 5, 4, 9
DATA 9, 4, 8, 2, 7, 4, 9, 1, 3, 0

```

② 整数データの xx% 番目は $(xx\% - 1) * 3 + 1$ で先頭が求まります。

③ 整数は 2 バイトそれに区切りのカンマで 3 バイトずつ使われています。

バイト	123	456	789	101112
値	5,	3,	2,	7,0

【プログラム例】

```

INPUT xx% ————— xx% 番目の入力
OPEN "b:reidai1.dat" FOR INPUT AS #1 ————— ファイルのオープン
SEEK #1, (xx% - 1) * 3 + 1 ————— ポインタを移動
INPUT #1, x% ————— 読み出し
CLOSE #1 ————— ファイルのクローズ
PRINT xx%; "ハンメ="; x% ————— 表示
END

```

【結 果】

? 17 17 を入力の場合
17 ハンメ= 9

【例題 168】 seek を使って浮動小数点型データを読み出す

次のデータをファイル reidai2.dat にかき、次に xx% 番目を入力して、xx% 番目のデータをよみ出すプログラムをつくれ。

データ; 4.3, 2.6, 3.5, 4.3, 5.4, 3.2, 2.7, 2.7, 8.9, 1.9

【解 説】 ◎ 浮動小数点型数値を seek を使って読み出します。

① 実数データをファイルへ書き込みます。

【プログラム例】

```

DIM x!(10)
FOR i = 0 TO 9
    READ x!(i)
NEXT i
OPEN "b:reidai2.dat" FOR OUTPUT AS #1
FOR i = 0 TO 9
    WRITE #1, x!(i)
NEXT i
CLOSE #1
END
DATA 4.3, 2.6, 3.5, 4.3, 5.4, 3.2, 2.7, 2.7, 8.9, 1.9

```

データの読み込み

データをファイルへ書き込み

② xx% 番目のデータは次の式で先頭が計算できます。

$$(xx\% - 1) * 5 + 1$$

③ 単精度実数は 4 バイト、それにカンマで、5 バイトが使われます。

バイト 1234567891011

値 4.3, 2.6, 3.5,1.9

【プログラム例】

```

INPUT xx%
OPEN "b:reidai2.dat" FOR INPUT AS #1
SEEK #1, (xx% - 1) * 5 + 1
INPUT #1, x!
CLOSE #1
PRINT xx%; "ハンメ="; x!
END

```

xx% 番目の入力

ファイルのオープン

ポインタを xx% 番目におく

読み出し

ファイルのクローズ

表示

【結 果】

? 3 3 を入力の場合
3 ハンメ= 3.5
? 10 10 を入力の場合
10 ハンメ= 1.9

【例題 169】 seek を使ってデータの修正

ファイル reidai2.dat を使い 3 番目のデータを 152.33 に変更するプログラムをつくれ。

【解 説】 ◎ seek を使ってデータの修正をします。

- ① ファイル reidai2.dat は浮動小数点単精度型のデータですから $x\%$ 番目のデータは $(x\%-1)*5+1$ へポインタをおいて、読み書きできます。ここでは、データを書き込みますので、前のデータが修正されます。

【プログラム例】

```
OPEN "b:reidai2.dat" FOR APPEND AS #1——— ファイルのオープン
SEEK #1, 11——— 3 番目にポインタをおく
WRITE #1, 152.33——— 152.33 をかく
CLOSE #1——— ファイルのクローズ
END
```

【結 果】 (【例題 168】のプログラムで読み出して確認)

```
? 3
3 ハンメ= 152.33
```

〔演習問題〕

- (174) reidai.dat のファイルを使い、6 番目を "ムロラン" に変更するプログラムをつくれ。
 (175) reidai.dat のファイルを使い、1 番目、3 番目、…9 番目と 1 つおきにデータを表示するプログラムをつくれ。
 (176) reidai2.dat のファイルを使い、2 番目、9 番目のデータの和を求めるプログラムをつくれ。

〔解 答〕**(174)【プログラム例】**

```
OPEN "b:reidai.dat" FOR APPEND AS #1——— ファイルのオープン
SEEK #1, 71——— 6 番目へポインタをおく
WRITE #1, "ムロラン"——— "ムロラン" をかく
CLOSE #1——— ファイルのクローズ
END
```

【結 果】

```
? 6
6 ハンメ=ムロラン
```

【例題 166】のプログラムで表示した例
6 番目が "ムロラン" となっている

(175) 【プログラム例】

```

OPEN "b:reidai.dat" FOR INPUT AS #1———ファイルのオープン
FOR i% = 1 TO 9 STEP 2———1 つおきに読み出して表示
    yy% = (i% - 1) * 14 + 1———1, 3, 5...9 番目へポインタをおく
    SEEK #1, yy%———
    INPUT #1, x$———読み出し
    PRINT x$———表示
NEXT i%
CLOSE #1———ファイルのクローズ
END

```

【結 果】

トウキョウ
 タカマツ
 チハ
 ヒロシマ
 トリデ

(176) 【プログラム例】

```

OPEN "b:reidai2.dat" FOR INPUT AS #1———ファイルのオープン
FOR i = 1 TO 2———2 番目と 9 番目のファイルデータを読んで和を求める
    READ xx%———何番目を読むかデータ文で指定
    yy% = (xx% - 1) * 5 + 1———ポインタを読み出すデータへおく
    SEEK #1, yy%———
    INPUT #1, x!———データの読み出し
    xx! = xx! + x!———和
NEXT i
PRINT xx!———表示
CLOSE #1———ファイルのクローズ
END
DATA 2, 9

```

【結 果】

11.5

24 章 メタコマンド

REM \$INCLUDE, \$STATIC, \$DYNAMIC

【例題 170】 インクルードファイル

変数 pi を 3.14159 とし、これをグローバル変数としてインクルードファイル "pai.in" で保存し、次にこれを使って sin (20 度) の値を求めるプログラムをつくれ。

【解 説】 ◎ メタコマンドでインクルードファイルを取り込みます。

- ① **GRPH**, **F**, **C**, ファイル名 "pai.in", 形式 "I/インクルード" で初期画面としてインクルードファイル "pai.in" を次のようにつくります。

【プログラム例】

```
COMMON pi
pi = 3.14159
```

END をつけないこと。END をつけるとインクルードされたとき、そこで処理が終わってしまいます。

- ② 次にこれを **GRPH**, **F**, **S** で保存します。
 ③ 次に **GRPH**, **E**, **N** で新規に次のプログラムをつくります。

【プログラム例】

```
REM $INCLUDE: 'b:pai.in'——インクルードファイルのとりこみ
PRINT SIN(20 / 180 * pi)——結果の表示
END
```

- ④ これを **GRPH**, **F**, **A** ファイル名 RRR-87 で格納します。
 ⑤ これを実行します。インクルードファイルの pi の値 3.14159 がインクルードされて SIN (20/180*pi) が実行され表示されます。

【結 果】

.3420199 (RRR-87 を実行した結果)

【例題 171】 メタコマンド include

ファイル hai.in を使って、下のデータを配列 A\$(12) によみ、次にこのファイルをインクルードして 1 つおきに表示するプログラムをつくれ。

データ: Tokyo, Osaka, Nagoya, Fukuoka, Niigata, Sendai,
 Yokohama, Kawasaki, Sapporo, Okayama, Kochi, Miyazaki

【解 説】 ◎ メタコマンド\$include を使います。

- ① 次のファイル"hai.in"をつくり保存します。

【プログラム例】 ファイル名 hai.in

```

DIM a$(12)
COMMON a$()          配列のコモン宣言
FOR i = 0 TO 11       データの読み込み
    READ a$(i)
NEXT i
DATA Tokyo,Osaka,Nagoya,Fukuoka
DATA Niigata,Sendai,Yokohama,Kawasaki
DATA Sapporo,Okayama,Kochi,Miyazaki

```

- ② 次の形でファイル"hai.in"をとりこみます。 hai.in では、配列 A\$(12)を宣言し、それをグローバル変数としています。 したがって、このプログラムの先頭でこの宣言を行ったのと同じ効果を持ちます。

```
REM $INCLUDE: 'b:hai.in'
```

メタコマンド名

メタコマンド

REM 文のようにかく :をいれてファイル名を''で囲っておく。

- ③ include を使うことでインクルードファイルを共通にどのプログラムからでも使えます。〔演習問題〕(178)でもこのインクルードファイルをとりこみます。
- ④ インクルードファイルはプロシージャ (SUB や FUNCTION) を使えません。
- ⑤ インクルードファイルはテキストファイルです。

【プログラム例】

```

REM $INCLUDE: 'b:hai.in'  インクルードファイルのとりこみ
FOR i = 0 TO 10 STEP 2    結果の表示
    PRINT a$(i)
NEXT i
END

```

【結 果】

```

Tokyo
Nagoya
Niigata
Yokohama
Sapporo
Kochi

```


【例題 172】 インクルードファイル

次のインクルードファイルをつくり、ファイル名"kamoku.in"で保存し、次に科目番号を入力した後、その科目の 20 人の点数を入力してその表示と平均点を表示せよ。

インクルードファイル"kamoku.in"の内容

```

DIM a(20), bs(5)
COMMON a(), bs()          ファイル kamoku.in
FOR i = 1 TO 5
  READ bs(i)
NEXT i
DATA 英語, 数学, 国語
DATA 社会, 理科

```

b\$(1)に英語, b\$(2)に数学, …b\$(5)に理科を代入, 配列 a(20), b\$(5)を宣言, それぞれをグローバルとする。

【解 説】 ① インクルードファイル

① インクルードファイル"kamoku.in"をつくり保存します。手順は次のとおりです。

[GRPH], **[F]**, **[C]**, ファイル名を b:kamoku.in, 形式を I/インクルードとします。 **[J]** で初期画面として、問題のテキストをかきます。その後 **[GRPH]**, **[F]**, **[S]** で保存します。

② 次の形で"b:kamoku.in"のインクルードファイルを取りこみます。

```
REM $INCLUDE: 'b:kamoku.in'
```

【プログラム例】

```

REM $INCLUDE: 'b:kamoku.in'  インクルードファイルのとりこみ
INPUT "科目番号"; x         番号の入力
FOR i = 1 TO 20             データの入力
  INPUT a(i)
NEXT i
PRINT bs(x); "の点数"      表示
FOR i = 1 TO 20
  PRINT a(i);
  t = t + a(i)
NEXT i
PRINT
PRINT "平均点"; t / 20
END

```

入力画面

```

科目番号? 2
? 54
? 89
?

```

【結 果】

数学の点数

54 89 98 69 58 77 46 100 85 65 95 54 77 86 94 56 68 76 99 75

平均点 76.05

【例題 173】 \$STATIC 配列と \$DYNAMIC 配列

次のデータをダイナミック配列 a\$(15)に読み、次に 1 文字を入力してそれで始まるデータを表示するプログラムをつくれ。ただし、プログラムの終りで配列を消去するものとする。

Shinagawa, Osaki, Gotanda, Meguro, Ebisu
Shibuya, Harajuku, Yoyogi, Shinjuku, Shinokubo
Takatanobaba, Mejiro, Ikebukuro, Otsuka, Sugamo

【解 説】 ◎ メタコマンドによって \$STATIC 配列と \$DYNAMIC 配列を指定します。

- ① プログラムをコンパイルするときに記憶領域を確保する \$STATIC 配列とプログラムを実行するとき記憶領域を確保する \$DYNAMIC 配列に分けて管理することができます。
これによって、\$DYNAMIC 配列を含むプログラムは実行されていないとき、メモリを他に有効に使えます。特に文字列配列、複数次元の配列はメモリ使用量が多いので有効です。
- ② DIM a(50)のように定数が添字の配列、添字が 10 以下で暗黙に宣言された配列は \$STATIC 配列です。
- ③ DIM a(x)のように変数によって添字が決められる配列、最初に COMMON で宣言される配列は \$DYNAMIC 配列です。
- ④ ここではメタコマンド \$STATIC と \$DYNAMIC を使って使い分けをします。
- ⑤ ERASE a\$によって \$STATIC 配列では配列の各要素の値が"" (ヌル) となります。\$DYNAMIC 配列では配列が削除されます。
- ⑥ REDIM a\$(100)で \$DYNAMIC 配列の割りあてのやり直しをします。添字が前と変わってもかまいません。ただし、次元数を変えることはできません。\$STATIC 配列では配列が削除されたわけではありませんから再宣言はエラーとなります。

【プログラム例】

```

REM $DYNAMIC a$ _____ ダイナミック配列 a$(15)の宣言
DIM a$(15) _____
FOR i = 0 TO 14 _____ データの読み込み
    READ a$(i)
NEXT i _____
INPUT x$ _____ 1 文字の入力
FOR j = 0 TO 14 _____ データをよみ、x$で始まるデータを表示
    IF ASC(a$(j)) = ASC(x$) THEN PRINT a$(j)
NEXT j _____
ERASE a$ _____ 配列の消去
END
DATA Shinagawa, Osaki, Gotanda, Meguro, Ebisu
DATA Shibuya, Harajuku, Yoyogi, Shinjuku, Shinokubo
DATA Takatanobaba, Mejiro, Ikebukuro, Otsuka, Sugamo

```

【結 果】

```
? 0————— 0 を入力の場合
Osaki
Otsuka
? S————— S を入力の場合
Shinagawa
Shibuya
Shinjuku
Shinokubo
Sugamo
```

〔演習問題〕

- (177) インクルードファイル `pai.in` を使って、`cos (20 度)` の値を求めるプログラムをつくれ。
- (178) ファイル `"hai.in"` を使って、3 番目のデータを表示するプログラムをつくれ。
- (179) インクルードファイル `"kamoku.in"` を使って、科目番号を入力し、次にその科目の 20 人の点数を入力して、各人の点数と最高点を表示するプログラムをつくれ。
- (180) 配列 `a$(6,2)` を `$STATIC` に宣言して次のデータの中の最初の 6 組のデータを読み、表示した後配列を消去し、後の 4 データを読んで表示するプログラムをつくれ。

名 前	出身地
山内 一男	大 阪
大友 雄次	九 州
中山 真一	東 北
河野 洋二	北 海 道
渋谷 一	北 陸
川上 三郎	東 京
池山 努	中 部
名古屋泰二	四 国
馬場 等	東 北
中村 勇	東 京

- (181) 配列 `a$(10,2)` を `$DYNAMIC` 配列とし、(180) のデータ 10 組を読み込んだ後、配列を消去し、`a$(13,2)` を再宣言して再度 10 組のデータをよみさらに 3 組を入力して追加し全体を表示するプログラムをつくれ。ただし、最後に配列を消去するものとする。

〔解 答〕

(177) 【プログラム例】

REM \$INCLUDE: 'b:pai.in' —— "pai.in"は【例題170】で作成済みとします
 PRINT COS(20 / 180 * pi) —— インクルードファイルのとりこみ
 END —— 結果の表示

【結 果】

.9396927

(178) 【プログラム例】

REM \$INCLUDE: 'b:hai.in' —— インクルードファイルのとりこみ
 PRINT a\$(3) —— 表示
 END

【結 果】

Fukuoka

(179) 【プログラム例】

REM \$INCLUDE: 'b:kamoku.in' —— インクルードファイルのとりこみ
 max = 0 —— max を 0 とする.
 INPUT "科目番号"; x —— 科目番号の入力
 FOR i = 1 TO 20 —— 点数の入力と最高点を max に求める
 INPUT a(i)
 IF max < a(i) THEN max = a(i)
 NEXT i
 PRINT b\$(x); "の点数" —— 20 人分の点の表示
 FOR i = 1 TO 20
 PRINT a(i);
 NEXT i
 PRINT
 PRINT "最高点"; max —— 最高点の表示
 END

入力画面

科目番号? 5
 ? 85
 ? 65
 ?

【結 果】

理科の点数

85 65 28 95 46 78 55 87 91 84 66 75 29 97 75 64 58 84 70 90

最高点 97

(180) 【プログラム例】

```

REM $STATIC a$ ————— スタティック配列 a$(6,2)の宣言
DIM a$(6, 2) —————
FOR i = 0 TO 5 ————— 配列ヘデータを6個読み込む
    READ a$(i, 0), a$(i, 1)
NEXT i
FOR j = 0 TO 5 ————— 6個のデータの表示
    PRINT a$(j, 0); " "; a$(j, 1)
NEXT j
ERASE a$ ————— 配列の初期化
RESTORE 100 ————— データポインタを移す
FOR k = 0 TO 3 ————— 後半4つのデータ組を読み込む
    READ a$(k, 0), a$(k, 1)
NEXT k
FOR i = 0 TO 3 ————— 4データ組の表示
    PRINT a$(i, 0); " "; a$(i, 1)
NEXT i
END
DATA ヤマウチ カズ`オ, オオサカ, オオトモ ユウジ`, キュウシュウ, ナカヤマシンイチ, トウホク
DATA コウノ ヨウジ`, ホッカイトー, シブ`ヤ ハジ`メ, ホクリク, カワカミ サブ`ロウ, トウキョウ
100 DATA イケヤマ ツトム, チュウブ`, ナゴ`ヤ タイジ`, シコク, ハ`ハ` ヒトシ, トウホク, ナカムラ イサム, トウキョウ

```

【結 果】

```

ヤマウチ カズ`オ   オオサカ
オオトモ ユウシ`   キュウシュウ
ナカヤマ シンイチ トウホク
コウノ ヨウシ`   ホッカイトー
シブ`ヤ ハシ`メ   ホクリク
カワカミ サブ`ロウ トウキョウ
イケヤマ ツトム   チュウブ`
ナゴ`ヤ タイシ`   シコク
ハ`ハ` ヒトシ   トウホク
ナカムラ イサム   トウキョウ

```

(181) 【プログラム例】

```

REM $DYNAMIC a$ ————— ダイナミック配列 a$(10,2)の宣言
DIM a$(10, 2) —————
FOR i = 0 TO 9 ————— データを読み込む
    READ a$(i, 0), a$(i, 1)
NEXT i
ERASE a$ ————— 配列の消去
REDIM a$(13, 2) ————— 配列の再宣言
RESTORE ————— データポインタをもどす
FOR j = 0 TO 9 ————— データを読み込む
    READ a$(j, 0), a$(j, 1)
NEXT j
FOR k = 10 TO 12 ————— 10~12番目のデータを入力する
    INPUT a$(k, 0)
    INPUT a$(k, 1)
NEXT k

```

```

FOR i = 0 TO 12 ————— 0～12 番目のデータを表示
    PRINT a$(i, 0); " "; a$(i, 1)
NEXT i —————
ERASE a$ ————— 配列の消去
END
DATA ヤマウチ カス`オ, オオサカ, オオトモ ヌウジ, キュウシュウ, ナカヤマシンイチ, トウホク
DATA コウノ ヨウジ, ホッカイトー, シブ`ヤ ハジ`メ, ホクリク, カワカミ サブ`ロウ, トウキョウ
100 DATA イケヤマ ツトム, チュウブ`, ナコ`ヤ タイジ, シコク, ハ`ハ` ヒトシ, トウホク, ナカムラ イサム, トウキョウ

```

【結 果】

```

ヤマウチ カス`オ      オオサカ
オオトモ ヌウシ`      キュウシュウ      "ウチダキヨシ"以下"トウホク"までを人力の例
ナカヤマ シンイチ     トウホク
コウノ ヨウシ`       ホッカイトー
シブ`ヤ ハシ`メ      ホクリク
カワカミ サブ`ロウ    トウキョウ
イケヤマ ツトム      チュウブ`
ナコ`ヤ タイシ`      シコク
ハ`ハ` ヒトシ        トウホク
ナカムラ イサム       トウキョウ
ウチダ` キヨシ       シコク
ノタ` ヒロシ         ホッカイトー
スカ` クニオ          トウホク

```

25 章 再帰プログラム

【例題 174】 再帰プログラム

整数を入力し、その値の階乗を求めるプログラムをつくれ。

ただし、階乗を求める部分を FUNCTION とし、再帰プログラムを使うものとする。

【解 説】 ◎ 再帰プログラムをつくります。

- ① 再帰プログラムとはある処理の中で自分自身を呼び出してくり返していくプログラムです。当然終了の条件をうまくつくってやる必要があります。
- ② 再帰の考え方の例として階乗を考えます。4 の階乗(4!) (!は階乗の意味、BASIC の単精度型実数を意味する宣言文字ではありません) は $4 \times 3 \times 2 \times 1$ です。したがって $4 \times 3!$ となります。3! は $3 \times 2!$ となります。2! は $2 \times 1!$ となります。
- ③ 一般的に n の階乗は $n \times (n-1)!$ で求められることがわかります。
- ④ $n!$ を求めるために $(n-1)$ の階乗を求めるプロシジユアを使い、 $(n-1)!$ を求めるために $(n-2)!$ を求めるプロシジユアを使います。このように階乗を求めるプロシジユアを自分自身の中で呼んで使います。
- ⑤ 次の形で再帰を使います。

```
FUNCTION kaijou (s%)
    IF S%=0 THEN
        kaijou=1
    ELSE
        kaijou=s%*kaijou(s%-1)
    ENDIF
END FUNCTION
```

入力された値が S% に引きわたされます。例として 4 とします。最初は S% が 0 ではありませんから $kaijou=4 \times kaijou(4-1)$ となります。

ここで $kaijou(4-1)$ が呼び出されます。S% が 3 として呼び出されます。最初は S% は 0 ではありませんから $kaijou=3 \times kaijou(3-1)$ です。

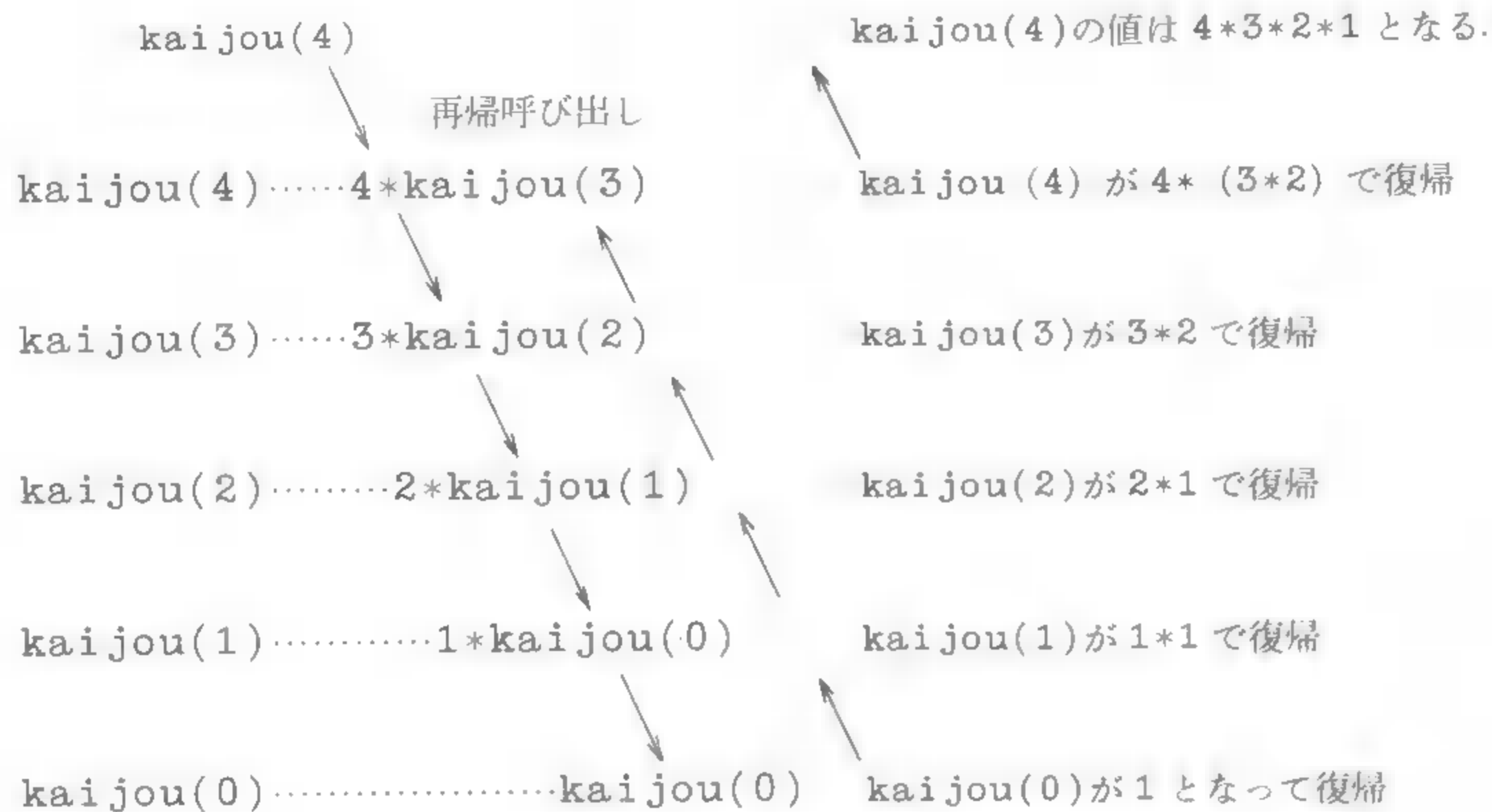
ここで $kaijou(3-1)$ が呼び出されます。S% が 2 として呼び出されます。最初は S% は 0 ではありませんから $kaijou=2 \times kaijou(2-1)$ です。

ここで $kaijou(2-1)$ が呼び出されます。S% が 1 として呼び出されます。最初は S% は 0 ではありませんから $kaijou=1 \times kaijou(1-1)$ です。

ここで $kaijou(1-1)$ が呼び出されます。S% が 0 として呼び出されます。S% が 0 ですから、

kaijou の値は 1 としてプロシジユアから復帰します。

すなわち $kaijou(1-1)$ は 1 です。したがって $kaijou=1*kaijou(1-1)$ は 1 となります。この値で復帰します。すると $kaijou(2-1)$ は 1 ですから $kaijou=2*kaijou(2-1)$ によりこれは 2 です。これで復帰すると $kaijou=3*kaijou(3-1)$ によりこの値は $3*2$ で 6 です。したがってこれで復帰すると $kaijou=4*kaijou(4-1)$ により、この値は $4*6$ で 24 です。この値が $kaijou$ の値となりメインにもどります。



【プログラム例】

```

DECLARE FUNCTION kaijou (s%)
INPUT seisu% ----- 整数の入力
PRINT seisu%; "!="; kaijou(seisu%) ----- kaijou を呼び出し結果を表示
END

FUNCTION kaijou (s%) ----- 階乗を求める FUNCTION
  IF s% = 0 THEN ----- S% が 0 のとき kaijou を 1 として終了
    kaijou = 1
  ELSE ----- kaijou の再帰呼び出し
    kaijou = s% * kaijou(s% - 1)
  END IF
END FUNCTION

```

【結 果】

? 5	5 を入力の場合
5 != 120	
? 10	10 を入力の場合
10 != 3628800	
? 15	15 を入力の場合
15 != 1.307674E+12	

〔演習問題〕

- (182) 整数を1つ入力して、1からその値まで和を求めるプログラムをつくれ。
- (183) 整数を2つ入力して、整数1の中から整数2をとり出す順列を求めるプログラムをつくれ。ただし、順列は整数1の階乗/(整数1-整数2)の階乗である。
- (184) 整数を2つ入力して、整数1の中から整数2をとり出す組合わせの数を求めるプログラムをつくれ。ただし、組合わせ数は整数1の階乗/(整数1-整数2)の階乗/整数2の階乗である。

〔解答〕

(182) 【プログラム例】

```

DECLARE FUNCTION wa (s%)
INPUT seisu% ————— 整数の入力
PRINT "1 から "; seisu%; "マデ`ノ`ワ`="; wa(seisu%) ——— 表示, wa を呼び出す
END

```

```

FUNCTION wa (s%) ————— FUNCTION wa
  IF s% = 0 THEN ————— 終了条件
    wa = 0
  ELSE
    wa = s% + wa(s% - 1) ————— 再帰呼び出し
  END IF
END FUNCTION

```

【結果】

```

? 10                                10 を入力の例
1 から 10 マデ`ノ`ワ` = 55
? 25                                25 を入力の例
1 から 25 マデ`ノ`ワ` = 325

```

(183) 【プログラム例】

```

DECLARE FUNCTION junretsu (s%)
INPUT seisu1% ————— 整数を2つ入力
INPUT seisu2% —————
ku1% = junretsu(seisu1%) ————— 整数 seisu1 の階乗
ku2% = junretsu(seisu1% - seisu2%) ——— 整数1-整数 seisu2 の階乗
PRINT seisu1%; "ノ`ナ`から "; seisu2% ——— 結果の表示
PRINT "オ`トリダ`ス`ジュンレツ`="; ku1% / ku2%
END

```

```

FUNCTION junretsu (s%) ————— 階乗を求める FUNCTION
  IF s% = 0 THEN ————— 終了条件
    junretsu = 1
  END IF
END FUNCTION

```

```

ELSE
    junretsu = s% * junretsu(s% - 1) ——再帰呼び出し
END IF
END FUNCTION

```

【結 果】

```

? 3                                     3 と 2 を入力の場合
? 2
3 ノ ナカカラ 2 オ トリダス シェンレツ = 6
? 5                                     5 と 3 を入力の場合
? 3
5 ノ ナカカラ 3 オ トリダス シェンレツ = 60
? 7                                     7 と 4 を入力の場合
? 4
7 ノ ナカカラ 4 オ トリダス シェンレツ = 840
? 10                                    10 と 8 を入力の場合
? 8
10 ノ ナカカラ 8 オ トリダス シェンレツ = 12160

```

(184) 【プログラム例】

```

DECLARE FUNCTION kumiawase (s%)
INPUT seisu1% ——整数を2つ入力
INPUT seisu2% ——
ku1% = kumiawase(seisu1%) ——整数1の階乗
ku2% = kumiawase(seisu1% - seisu2%) ——(整数1 - 整数2)の階乗
ku3% = kumiawase(seisu2%) ——整数2の階乗
PRINT seisu1%; "ノ ナカカラ "; seisu2% ——結果の表示
PRINT "オ トリダス クミアワセ = "; ku1% / ku2% / ku3% ——
END

FUNCTION kumiawase (s%) ——階乗を求める FUNCTION
IF s% = 0 THEN ——終了条件
    kumiawase = 1 ——再帰呼び出し
ELSE
    kumiawase = s% * kumiawase(s% - 1)
END IF
END FUNCTION

```

【結 果】

```

? 7                                     7 と 2 を入力の場合
? 2
7 ノ ナカカラ 2 オ トリダス クミアワセ = 21
? 6                                     6 と 3 を入力の場合
? 3
6 ノ ナカカラ 3 オ トリダス クミアワセ = 20

```

26 章 バージョン 4.5 で追加された命令

SOUND, SLEEP, ストップ・キー割込み, ヘルプ・キー割込み

【例題 175】 音 (Ver. 4.5 以降の機能)

ドレミファソラシド を鳴らすプログラムをつくれ。

【解 説】 ◎ SOUND は指定した周波数の音を指定した長さだけ鳴らします。

① SOUND x, y は周波数 x の音を y 時間の間だけ鳴らします。

SOUND x, y

0 から 65,535 の値

0 はオフで 18.2 で 1 秒となる。

周波数

37 から 32,761 までの値

② 音階をつくってみます。

ドレミファソラシドは下のドを 1, 上のドを 2 としてその間を下のように分けたものです。下のドは周波数で約 520 ヘルツです。したがって, x を 520, y を 9.1 とすると下のドを 0.5 秒鳴らします。

ド	レ	ミ	ファ	ソ	ラ	シ	ド
1	$6\sqrt{2}$	$3\sqrt{2}$	$12\sqrt{2^5}$	$12\sqrt{2^7}$	$4\sqrt{2^3}$	$12\sqrt{2^{11}}$	2

③ この機能は 9801E/F/M/U では使えません。

【プログラム例】

```

xx = 520: y = 9.1
FOR i = 1 TO 8
  READ x _____ 音階を示す
  SOUND x * xx, y _____ 周波数と 0.5 秒を示す。
NEXT i
END
DATA 1, 1.1225, 1.2599, 1.3348 _____ 音階データ
DATA 1.4983, 1.6818, 1.8877, 2

```

【結 果】

ドレミファソラシドが鳴ります。

【例題 176】 1 時停止 (Ver. 4.5 以降の機能)

"1 ビョウゴニヒョウジ"と表示して、1 秒後に 1, "2 ビョウゴニヒョウジ"と表示して 2 秒後に 2 を表示、以下同様に "5 ビョウゴニヒョウジ"と表示して 5 秒後に 5 と表示するプログラムをつくれ。

【解 説】 ◎ SLEEP は 1 時停止します。

① SLEEP x は x 秒間コンピュータを一時停止します。再開する条件は次のとおりです。x が省略されると再開の条件は次の (2) と (3) です。

- (1) x 秒の経過。
- (2) 何かキーがおされたとき。
- (3) 再開の割込み処理が行われたとき。

【プログラム例】

```
CLS
FOR i = 1 TO 5
  PRINT i; "ビョウゴニヒョウジ"
  SLEEP i
  PRINT i
NEXT i
END
```

1~5 のくり返し
i ビョウゴニヒョウジ の表示
i 秒停止
i の表示

【結 果】

```
1 ビョウゴニヒョウジ
1
2 ビョウゴニヒョウジ
2
3 ビョウゴニヒョウジ
3
4 ビョウゴニヒョウジ
4
5 ビョウゴニヒョウジ
5
```

【例題 177】 ストップキー割込み (Ver. 4.5 以降の機能)

1~1000 のくり返しを FOR~NEXT 文でかき、その間にストップキーがおされたらその時の i を表示するプログラムをつくれ。

【解 説】 ◎ ストップキーによる割込みをします。

① 次の形でストップキー割込みをします。

```

      ┌ ストップ・キーを示す ─┐ 割込み処理サブルーチン名
      │                         │
      └──────────────────┘
ON KEY(30) GOSUB ラベル ─────────── 割込み
KEY(30) ON ─────────── 割込みオン

```

END

ラベル; _____ 割込み処理ルーチン
/
RETURN _____

【プログラム例】

```
ON KEY(30) GOSUB stopkey _____ ストップ・キー割込み
KEY(30) ON _____ 割込みオン
CLS
FOR i = 1 TO 10000: NEXT i _____ くり返し
END
stopkey: _____ 割込み処理ルーチン
PRINT i;
RETURN _____
```

【結 果】

2832 5086 8077 途中で3回 **STOP** キーをおした例

【例題 178】 ヘルプキー割込み (Ver. 4.5 以降の機能)

1~1000 のくり返しを FOR~NEXT 文でかき、その間にヘルプキーがおされたら、そのときの値とその平方根を表示するプログラムをつくれ。

【解 説】 ㊦ ヘルプキーによる割込みをします。

① 次の形でヘルプ・キー割込みをします。

ヘルプ・キーを示す _____ 割込み処理サブルーチン名

```
ON KEY(31) GOSUB ラベル _____ 割込み
KEY(31) ON _____ 割込みオン
/
END
ラベル; _____ 割込み処理ルーチン
/
RETURN _____
```

【プログラム例】

```
ON KEY(31) GOSUB helpkey _____ ヘルプキー割込み
KEY(31) ON _____ 割込みオン
CLS
FOR i = 1 TO 10000: NEXT i _____ くり返し
END
helpkey: _____ 割込み処理
PRINT i; SQR(i);
RETURN _____
```

【結 果】

2683 51.79768 5033 70.94364 9014 94.94209 3回 **HELP** キーをおした例。

III.....グラフィック編



バージョン 4.5 では環境変数 QBGRPNEC=YES のセット (A>SET QBGRPNEC=YES)により、グラフィック画面は MS-DOS にもどうしてもクリアしません。

バージョン 4.2 ではクリアしますのでグラフィック画面は MS-DOS にもどすときに消えてしまいます。

1 章 直線, 四角形, 円, 楕円, 扇形, 点

SCREEN, CLS, LINE, LINE STEP, LINE~B (F) CIRCLE,
PSET, PRESET

【例題 1】 直線の表示

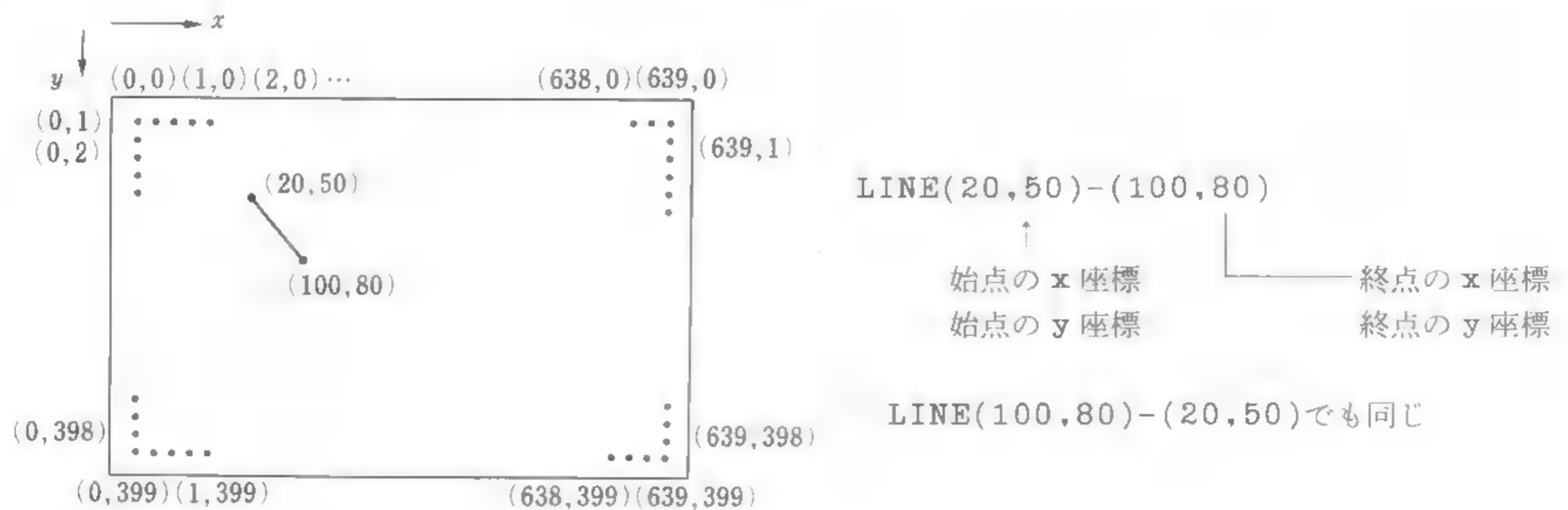
画面左上から右下へ斜め斜線をかくプログラムをつくれ。

【解 説】 ◎ 直線をかきます。

- ① SCREEN 文は画面のモードを設定します。
- ② SCREEN 0 または SCREEN 88 は横 640 ドット*縦 400 ドットのグラフィック画面とテキスト画面を 2 頁用意します。(PC9801/U の場合は 1 頁) モードは次のようなものがあります。

モード	種 類	グラフィック解像度
81	テキスト	(80 桁*25 桁)
84	スーパーインポーズ	640*200
87	グラフィック	640*400
88, 0	スーパーインポーズ	640*400

- ③ CLS 0 はグラフィック画面をクリアーします。
- ④ 直線は下図の座標系の中で指定された 2 つの位置を直線でむすびます。



- ⑤ グラフィックポインタがかき終った (100, 80) におかれます。

【プログラム例】

```
SCREEN 0 画面設定
CLS 0 クリア
LINE (20, 50)-(100, 80) 直線
END
```

【結 果】

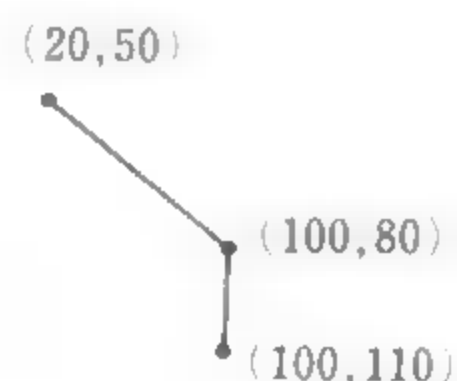
【例題 2】 連続した直線

(20, 50) と (100, 80) と (100, 110) をむすび、次に (100, 110) から (-20, +30) の点とそこから (+50, +20) の点をむすび、最後にそこから (+50, 0) の点をむすぶ図をつくるプログラムをつくれ。

【解 説】 ◎ 連続した直線をかきます。絶対座標と相対座標の表わし方を示します。

- ① 次の形で (20, 50) と (100, 80) および (100, 110) を直線で結びます。各々の点の座標は絶対座標値で表わされています。

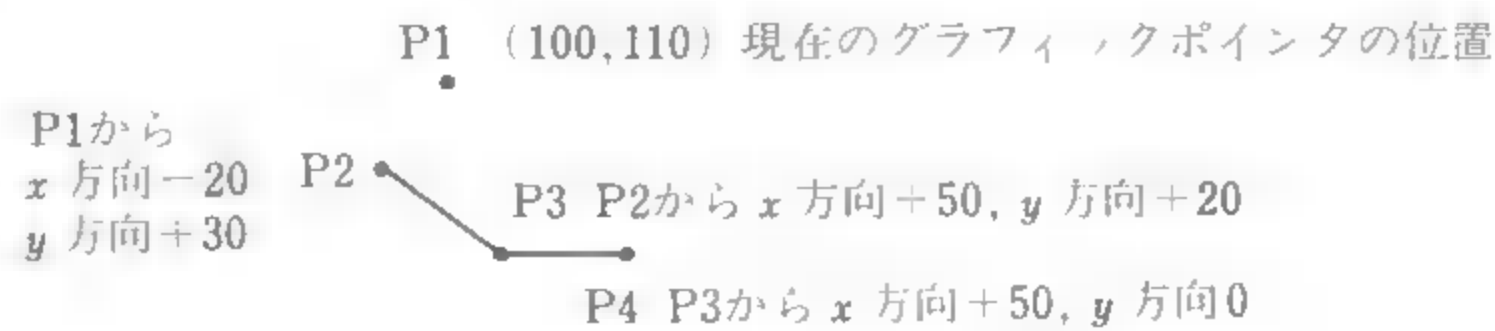
```
LINE(20,50)-(100,80) (20,50)と(100,80)を直線で結ぶ
LINE-(100,110) 現在グラフィックポインタのある(100,80)と(100,110)を結ぶ
```



- ② 現在のグラフィックポインタを基準にして相対値を与えて直線をひくことができます。

```
LINE STEP(-20,30)-STEP(50,20) 現在のグラフィックポインタから
x 方向-20, y 方向+30 の位置と、
そこから x 方向+50, y 方向+20
の位置をむすぶ。
```

LINE-STEP(50,0) ———— さらにそこから x 方向+50, y 方向 0 の位置をむすぶ



【プログラム例】

```
SCREEN 0
CLS 0
LINE (20, 50)-(100, 80)
LINE -(100, 110)
LINE STEP(-20, 30)-STEP(50, 20)
LINE -STEP(50, 0)
END
```

【結 果】




【例題3】 箱

四角形と、中を塗りつぶした四角形をかくプログラムをつくれ。

【解 説】 ◎ 箱と中を塗りつぶした四角形をかきます。

① 次の形で箱（四角形）をかきます。

LINE(20,50)-(100,80),,B

(20,50)  (100,80)

LINE(20,50)-(100,80),,B
 (20,50)と(100,80)を対角とする四角. グラフィックポイントは(100,80)におかれます.
 LINE(100,80)-(20,50),,Bでも同じ
 LINE(100, 50)-(20, 80),,Bでも同じ
 LINE(20, 80)-(100, 50),,Bでも同じ } グラフィックポイントは後の点となります。

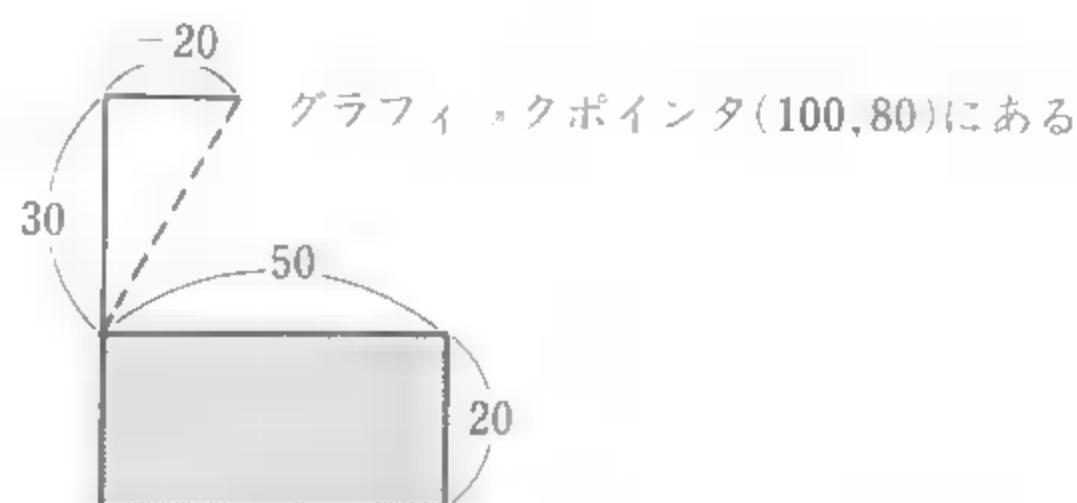
② x 方向の増分と y 方向の増分が同じとき正方向となります。

LINE(20,50)-(100,130),,B は正方形となります。



- ③ 箱の場合も STEP を使って、位置を相対的に与えることができます。また B を BF とすると箱の中を塗りつぶします。

```
LINE STEP(-20,30)-STEP(50,20),,BF
```



BFで中を塗りつぶす

(100,80)から x 方向-20, y 方向+30 の点とそこから x 方向+50, y 方向+20 の点を対角とする箱をかき中を塗ります

【プログラム例】

```
SCREEN 0
CLS 0
LINE (20, 50)-(100, 80),, B
LINE STEP(-20, 30)-STEP(50, 20),, BF
END
```

【結果】



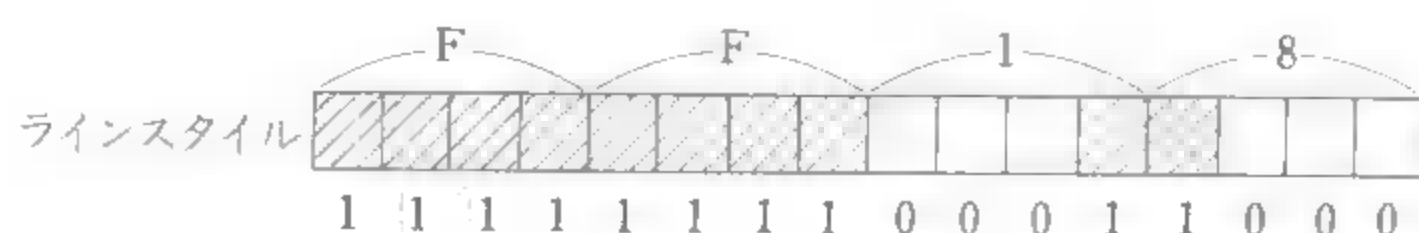
【例題 4】 ラインの指定

点線と 1 点鎖線で四角形をかくプログラムをつくれ。

【解説】 ◎ ラインオプションで線の種類を指定できます。

- ① LINE 文の 3 番目のオプションにラインスタイルをつけることができます。ラインスタイルは次のように決めます。

LINE(20,100)-(100,130),,B,&HFF18 の例



図

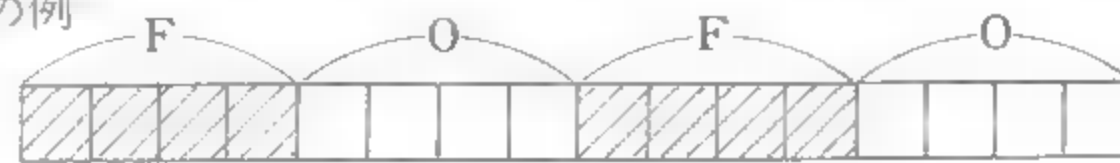


16 ビットを 4 ビットずつに区切って 16 進数で表わす。

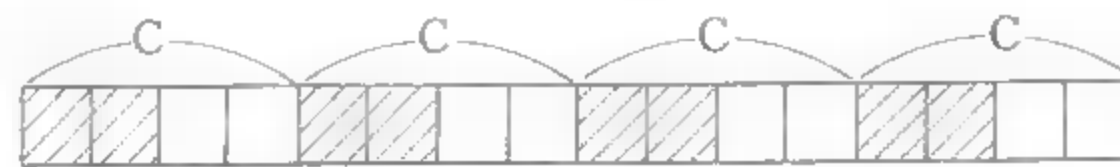
この例では &HFF18 となる。このパターンのくり返しとなる。したがって 1 点鎖線となる。

- ② 点線は次のようになります。

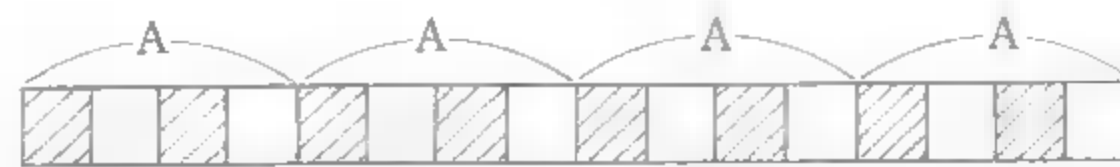
点線の例



大きい点線



中位の点線



細かい点線

【プログラム例】

```
SCREEN 0
CLS 0
LINE (20, 50)-(100, 80), , B, &HF0F0
LINE (20, 100)-(100, 130), , B, &HFF18
END
```

【結果】



【例題5】 円

中心を (100, 100) とし、半径を 40 とする円をかくプログラムをつくれ。

【解説】 ◎ 円をかきます。

① 円は次のようにかきます。

```
CIRCLE(100, 100), 40
```

円 中心の X 座標 半径
 中心の Y 座標

【プログラム例】

```
SCREEN 0
CLS 0
CIRCLE (100, 100), 40
END
```

【結果】



【例題 6】 半円と中心位置を相対指定

中心を (100, 100)、半径を 40 とする上半分の半円と、中心を x 方向に+200 した位置に半径 50 の円をかくプログラムをつくれ。

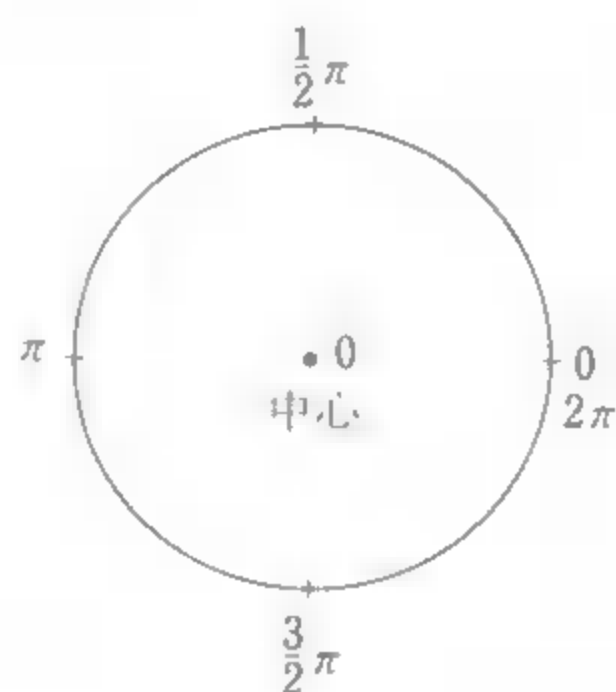
【解 説】 ◎ 半円と中心を相対指定します。

① 次の形で半円をかきます。

`CIRCLE(100, 100), 40, , 0, 3.14159`

中心座標 半径 開始点 終了点

0 から π まで反時計方向でかく



開始点と終了点の表わし方
0～2 ラジアンで表わす。
表示は反時計方向

② 中心座標は相対的に表わすことができます。

`CIRCLE STEP(200, 0), 50`

グラフィックポインタ (本例では 100, 100) から
x 方向に+200、y 方向は 0 の点を中心とする。
円のグラフィックポインタは中心点です。

【プログラム例】

```
SCREEN 0
CLS 0
CIRCLE (100, 100), 40, , 0, 3.14159
CIRCLE STEP(200, 0), 50
END
```

【結 果】



【例題 7】 扇形

結果の扇形をかくプログラムをつくれ。

【解 説】 ◎ 扇形をかきます。

- ① 開始点と終了点にマイナスをつけると中心と開始点、中心と終了点を直線で結びます。したがって扇形となります。

CIRCLE (100, 100), 40, , -.5, -2



【プログラム例】

```
SCREEN 0
CLS 0
CIRCLE (100, 100), 40, , -.5, -2
END
```

【結果】



【例題8】 楕円

結果の楕円を2つかくプログラムをつくれ。

【解説】 ◎楕円をかきます。

- ① CIRCLE 文の最後のオプションの値で円を変形して楕円とします。

CIRCLE STEP(200, 0), 50, , , , .5

(0, 0)からの相対指定

長軸半径

横長の楕円

1未満のとき横長の楕円. 1が円

CIRCLE STEP(200, 0), 50, , , , 1.5

(200, 0)からの
相対指定

長軸半径

縦長の楕円

1をこえるとき縦長の楕円

【プログラム例】

```
SCREEN 0
CLS 0
CIRCLE (200, 100), 50, , , , .5
CIRCLE STEP(200, 0), 50, , , , 1.5
END
```

【結果】

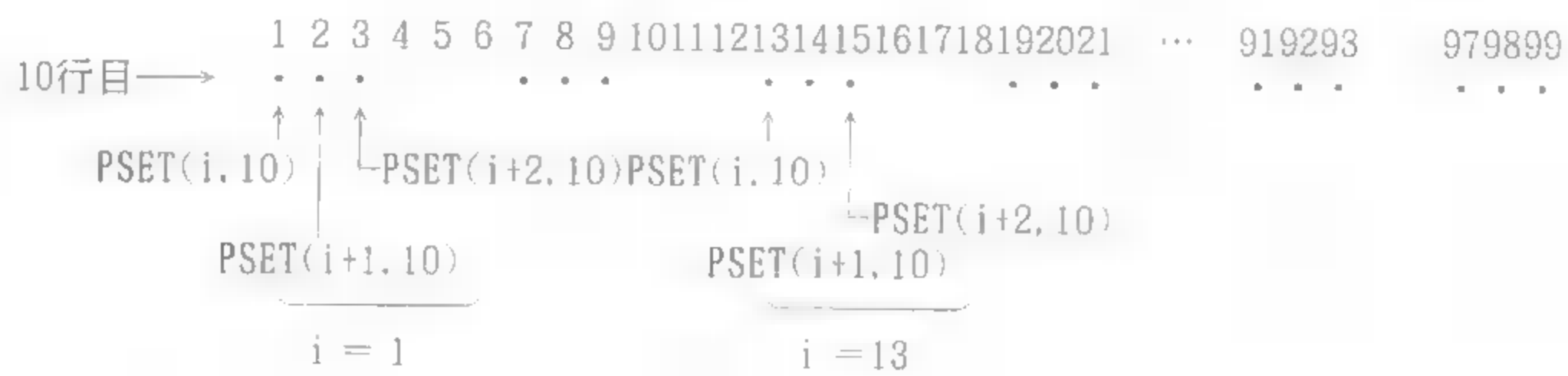


【例題 9】 ドットの表示

1 ドットを表示することのくり返しで点線にかくプログラムをつくれ。

【解 説】 ◎ ドットの表示をします。

- ① PSET は 1 ドットを表示します。表示位置は (x,y) で指定します。
- ② 10 ドット行目を指定して、i を 1 から 100 まで増分を 6 でくり返し、i, i+1, i+2 を表示し、i+3, i+4, i+5 を非表示とすると点線となります。



【プログラム例】

```
SCREEN 0
CLS 0
FOR i = 1 TO 100 STEP 6
    PSET (i, 10)
    PSET (i + 1, 10)
    PSET (i + 2, 10)
NEXT i
END
```

【結 果】

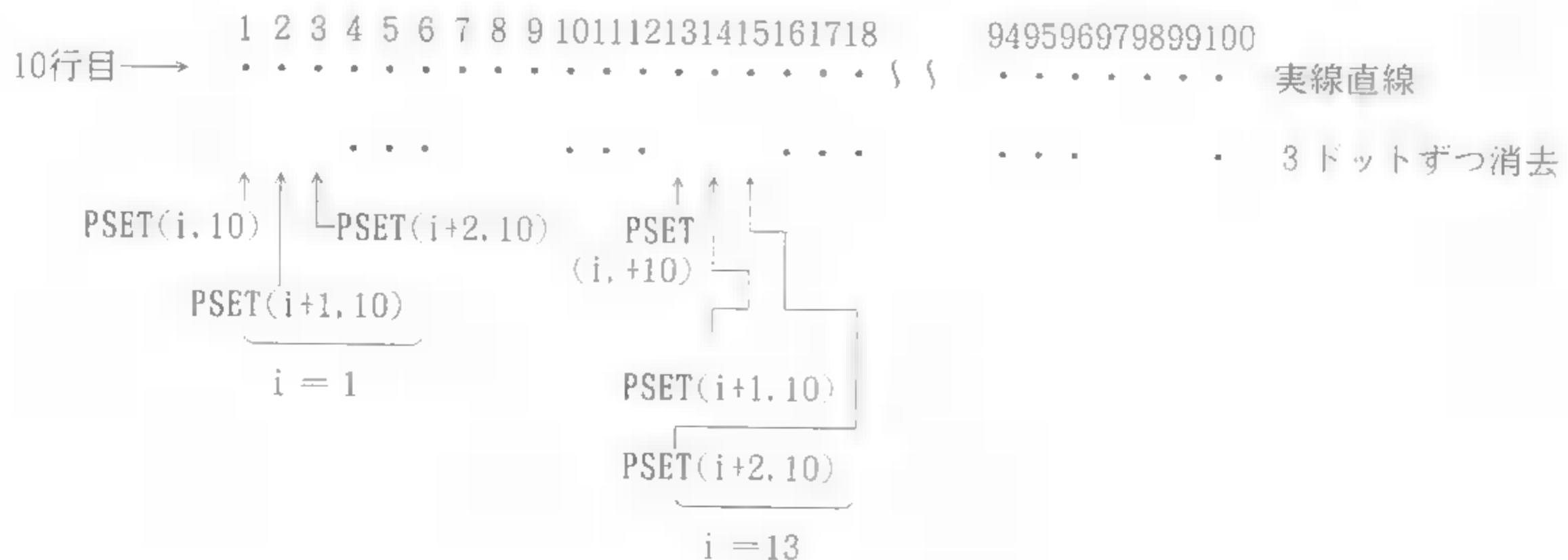
.....

【例題 10】 ドットの表示を非表示とする

横線を実線でかき、次にそれを点線にかえるプログラムをつくれ。

【解 説】 ◎ 表示されているドットを非表示とします。

- ① PRESET(x,y)で(x,y)のドットを非表示とします。
- ② (1,10)と(100,10)をむすぶ実線をかき、3 ドットずつ消去すると点線となります。



【プログラム例】

```

SCREEN 0
CLS 0
LINE (1, 10)-(100, 10)
FOR i = 1 TO 100 STEP 6
    PRESET (i, 10)
    PRESET (i + 1, 10)
    PRESET (i + 2, 10)
NEXT i
END

```

【結 果】

.....

〔演習問題〕

- (1) ひし形をかくプログラムをつくれ.
- (2) 結果の図をつくるプログラムをつくれ.
- (3) 点線の円をかくプログラムをつくれ.
- (4) 1点鎖線の円をかくプログラムをつくれ.
- (5) 結果の図をつくるプログラムをつくれ.
- (6) 結果の図をつくるプログラムをつくれ.
- (7) 中心を (300, 200) とし、半径を 20, 40, 60, 80, 100 の同心円をかくプログラムをつくれ.
- (8) 結果の図をかくプログラムをつくれ.

〔解 答〕

(1) 【プログラム例】

```

SCREEN 0
CLS 0
LINE (100, 100)-(130, 150)
LINE -(100, 200)
LINE -(70, 150)
LINE -(100, 100)
END

```

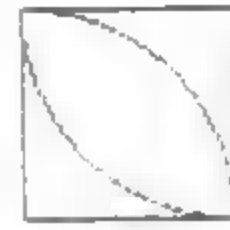
【結 果】



(2) 【プログラム例】

```
SCREEN 0
CLS 0
LINE (100, 100)-(150, 150), , B
CIRCLE (100, 150), 50, , 0, 3.14159 / 2
CIRCLE (150, 100), 50, , 3.14159, 1.5 * 3.14159
END
```

【結果】



(3) 【プログラム例】

```
SCREEN 0
CLS 0
FOR i = 0 TO 2 * 2.94159 STEP .4
    CIRCLE (100, 150), 40, , i, i + .2
NEXT i
END
```

【結果】



(4) 【プログラム例】

```
SCREEN 0
CLS 0
FOR i = 0 TO 2 * 2.89159 STEP .5
    CIRCLE (100, 150), 40, , i, i + .2
    CIRCLE (100, 150), 40, , i + .35, i + .35
NEXT i
END
```

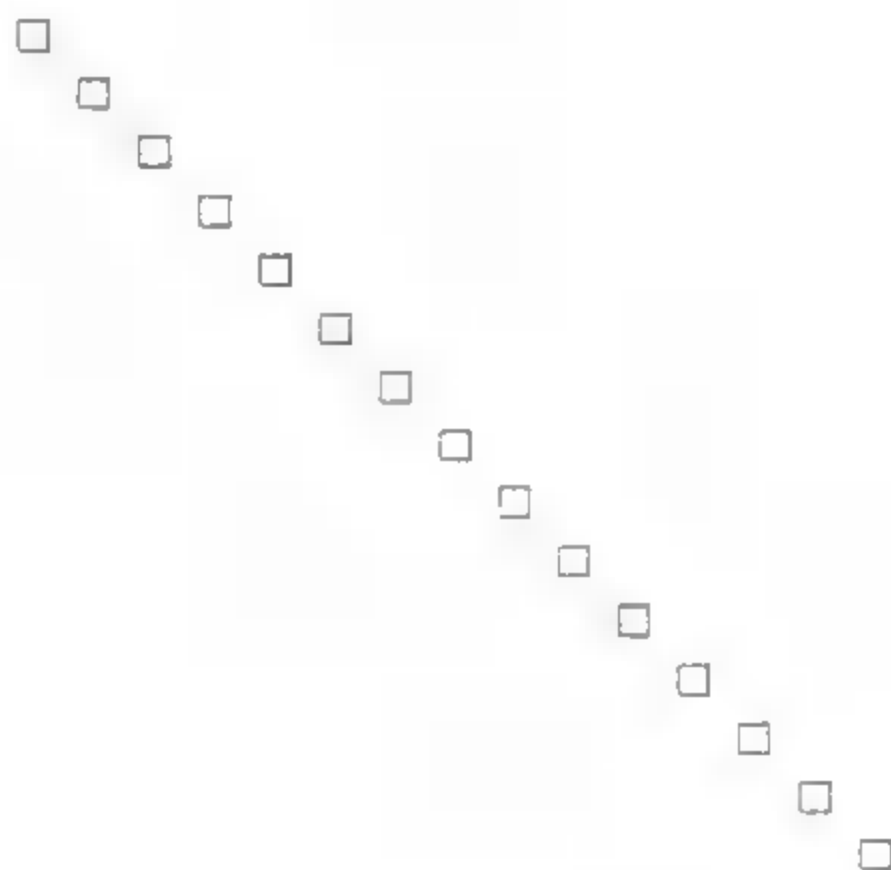
【結果】



(5) 【プログラム例】

```
SCREEN 0
CLS 0
FOR i = 1 TO 15
    LINE (i * 20, i * 20)-(i * 20 + 10, i * 20 + 10), , B
NEXT i
END
```

【結果】



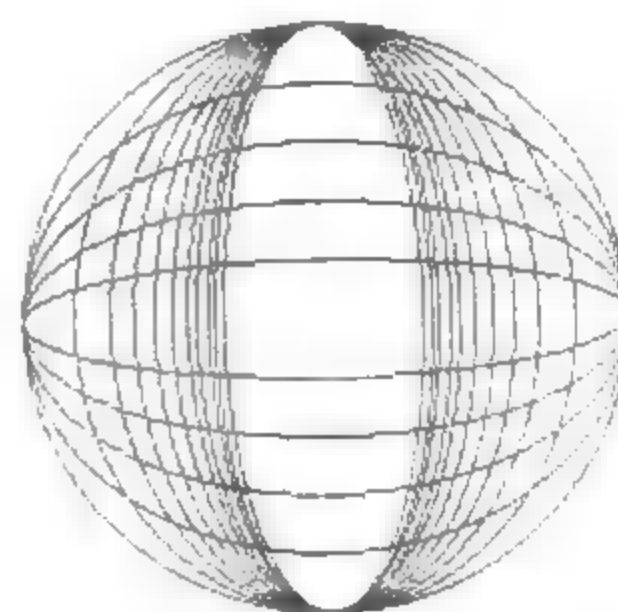
(6) 【プログラム例】

```

SCREEN 0
CLS 0
FOR i = 1 TO 20
    CIRCLE (300, 200), 100, , , , .2 * i
NEXT i
END

```

【結果】



(7) 【プログラム例】

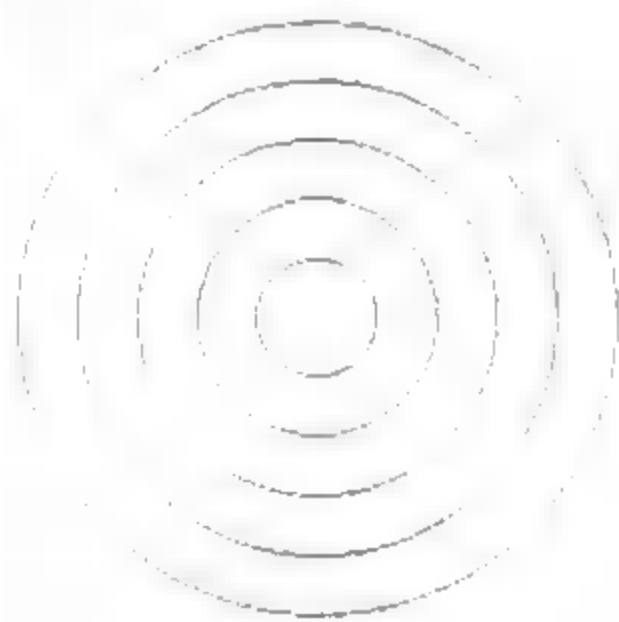
```

SCREEN 0
CLS 0
FOR i = 20 TO 100 STEP 20
    CIRCLE (300, 200), i
NEXT i
END

```

半径を 20, 40, 60, 80, 100 として同心円を表示

【結果】



(8) 【プログラム例】

```

SCREEN 0
CLS 0
CIRCLE (300, 200), 100, , -3.14159 * 1 / 8, -3.14159 * 7 / 8
CIRCLE (300, 200), 20, , -2 * 3.14159 * 9 / 16, -2 * 3.14159 * 15 / 16
CIRCLE (300, 150), 20
PAINT (300, 150), 7, 7
END

```

扇形
扇形
円
塗りつぶし

【結果】



2章 塗りつぶし;タイルパターン

PAINT

【例題 11】 塗りつぶし

三角形を塗りつぶすプログラムをつくれ.

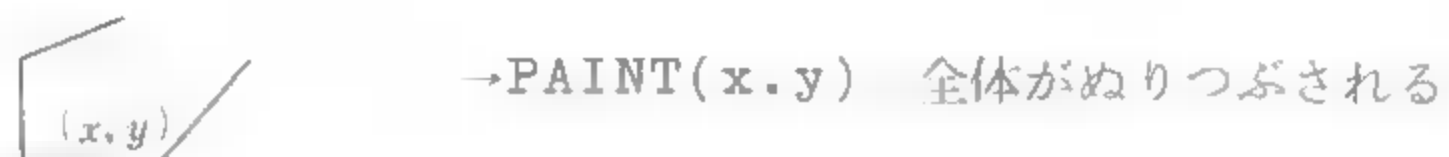
【解 説】 ◎ 塗りつぶしをします.

① 閉じた境界内を塗りつぶすことができます.

PAINT(x, y) で(x, y) を含む閉じた境界内を塗りつぶします.



② 閉じていないときは全体を塗りつぶします.



【プログラム例】

```
SCREEN 0
CLS 0
LINE (50, 10)-STEP(20, 30)
LINE -STEP(-40, 0)
LINE -STEP(20, -30)
PAINT (50, 20)
END
```

【結 果】



【例題 12】 タイルパターン

結果の 5 種類のタイルパターンで 5 つの長方形をうめるプログラムをつくれ.

【解 説】 ◎ タイルパターンをつくり、そのパターンで箱を埋めます。

① タイルパターンは次のようにつくりします。

CHR\$(&H0)+CHR\$(&H0)+CHR\$(&H0)+CHR\$(&HFF)
 下の 4 つの 8 ピクセルの状態の和が 1 行 8 ピクセルの状態

B 0 0 0 0 0 0 0 0 &H00 (B(青)の 8 のピクセルの状態)

G 0 0 0 0 0 0 0 0 &H00 (G(緑)の 8 のピクセルの状態)

R 0 0 0 0 0 0 0 0 &H00 (R(赤)の 8 のピクセルの状態)

輝度 1 1 1 1 1 1 1 1 &HFF (輝度の 8 のピクセルの状態)

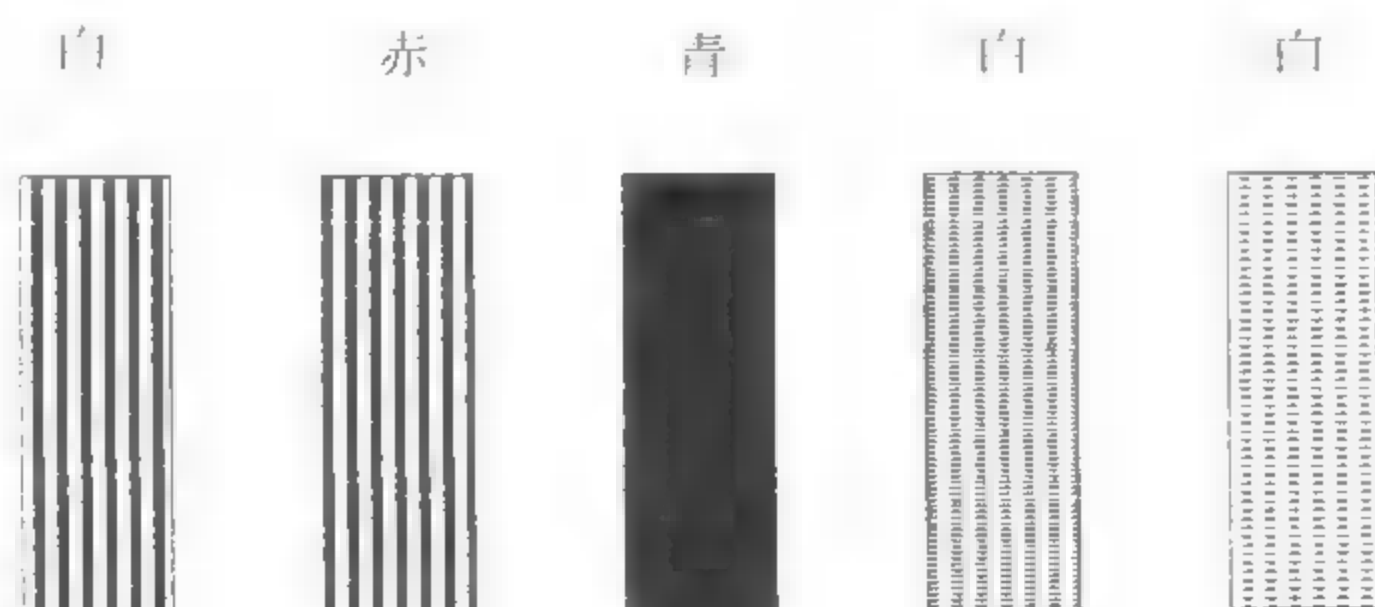
② 2 行のパターンをつくるときは 1 行分の CHR\$ の和の後に 4 つの CHR\$ の和を加えます。

③ 3行以上の場合も同様に1行につき4つのCHR\$の和をつなぎます。

【プログラム例】

```
SCREEN 0
CLS 0
FOR i = 1 TO 5
LINE (i * 100, 100)-(i * 100 + 50, 250), , B
NEXT i
t1$ = CHR$(&HFO) + CHR$(&HFO) + CHR$(&HFO) + CHR$(&HFF)
PAINT (120, 120), t1$, 7
t2$ = CHR$(&HO) + CHR$(&HO) + CHR$(&HFO) + CHR$(&HFF)
PAINT (220, 120), t2$, 7
t3$ = CHR$(&HFF) + CHR$(&HO) + CHR$(&HO) + CHR$(&HFF)
PAINT (320, 120), t3$, 7
t4$ = CHR$(&HO) + CHR$(&HO) + CHR$(&HO) + CHR$(&HO)
t5$ = t1$ + t4$
PAINT (420, 120), t5$, 7
t6$ = t5$ + t4$ + t4$ + t4$
PAINT (520, 120), t6$, 7
END
```

【結 果】



〔演習問題〕

- (9) 結果の図をつくるプログラムをつくれ.
- (10) 結果の図をつくるプログラムをつくれ.
- (11) 結果の5つのパターンで円をぬるプログラムをつくれ.

〔解 答〕

(9) 【プログラム例】

```
SCREEN 0
CLS 0
FOR i = 1 TO 10
    CIRCLE (300, 200), i * 10
    IF (i MOD 2) = 0 THEN PAINT (300 + i * 10 - 5, 200), 7, 7
NEXT i
END
```

【結 果】



(10) 【プログラム例】

```
SCREEN 0
CLS 0
LINE (100, 100)-(150, 150), , B
CIRCLE (100, 150), 50, , 0, 3.14159 / 2
CIRCLE (150, 100), 50, , 3.14159, 1.5 * 3.14159
PAINT (125, 125), 7, 7
END
```

【結 果】



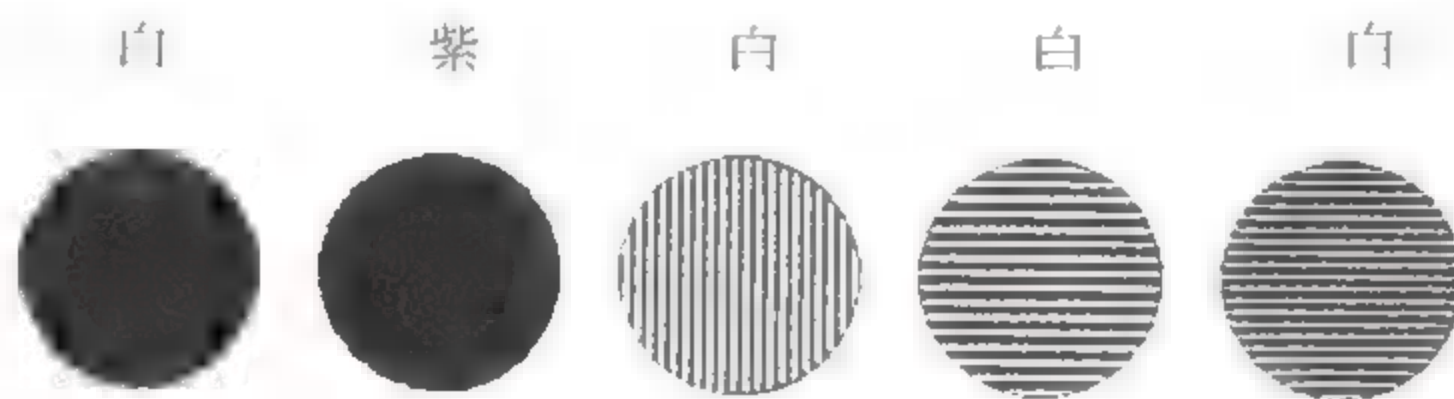
(11) 【プログラム例】

```

SCREEN 0
CLS 0
FOR i = 1 TO 5 ————— 5 つの円をかく
CIRCLE (i * 100, 100), 40
NEXT i
t1$ = CHR$(&HFF) + CHR$(&HFF) + CHR$(&HFF) + CHR$(&HFF) ———— タイルパターン 1
PAINT (100, 100), t1$, 7 ————— タイルパターン 1 で円をぬる
t2$ = CHR$(&HFF) + CHR$(&H0) + CHR$(&HFF) + CHR$(&HFF) ———— タイルパターン 2
PAINT (200, 100), t2$, 7 ————— タイルパターン 2 で円をぬる
t3$ = CHR$(&HCC) + CHR$(&HCC) + CHR$(&HCC) + CHR$(&HFF) ———— タイルパターン 3
PAINT (300, 100), t3$, 7 ————— タイルパターン 3 で円をぬる
t4$ = CHR$(&H0) + CHR$(&H0) + CHR$(&H0) + CHR$(&H0) ———— タイルパターン 4
t5$ = t1$ + t1$ + t4$ + t4$ ————— タイルパターン 5
PAINT (400, 100), t5$, 7 ————— タイルパターン 5 で円をぬる
t6$ = t1$ + t3$ + t4$ ————— タイルパターン 6
PAINT (500, 100), t6$, 7 ————— タイルパターン 6 で円をぬる
END

```

【結 果】



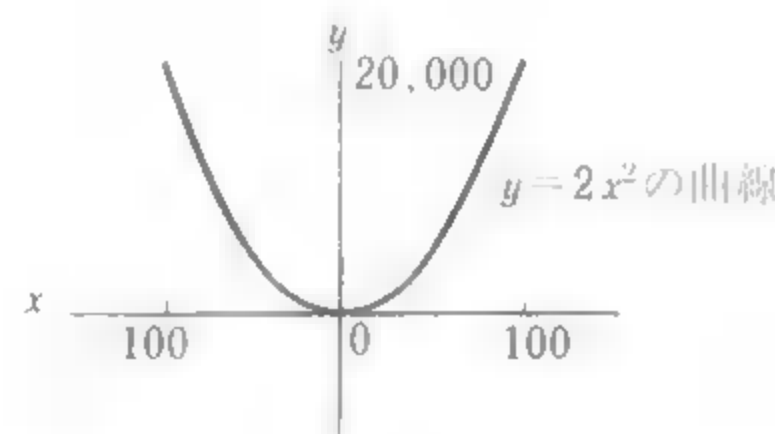
3 章 関数式の曲線

【例題 13】 関数式の表示

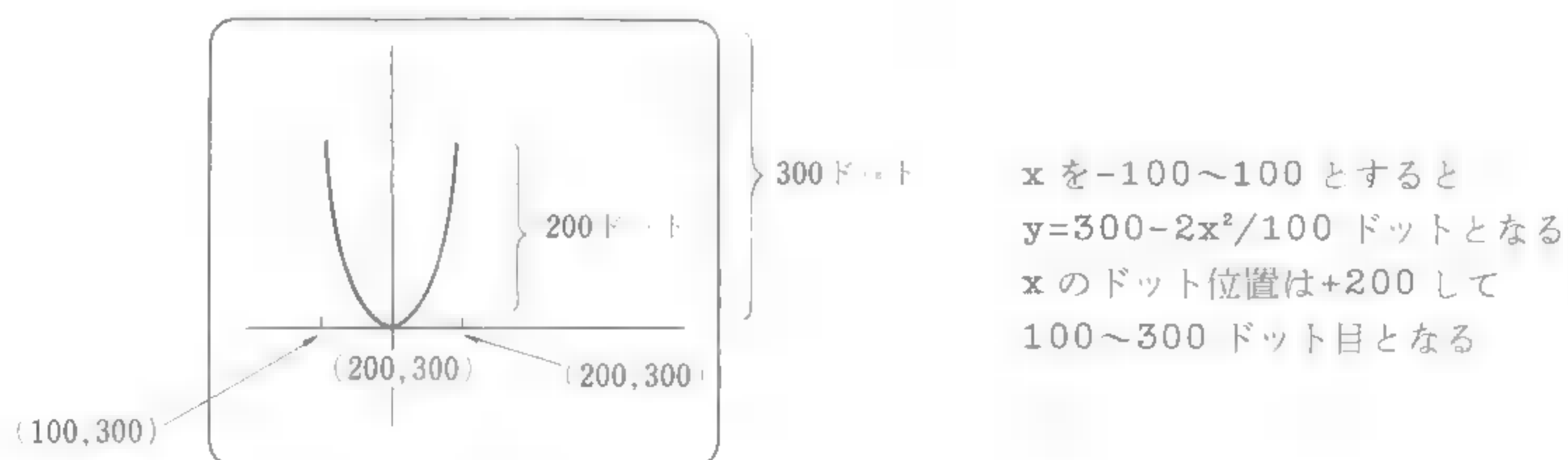
$y=2x^2$ の曲線を x を -100 から $+100$ までの値としたときの y の値の曲線を表示するプログラムをつくれ.

【解 説】 ◎ $y=2x^2$ の関数式の曲線を表示します.

- ① $y=2x^2$ の式で x を -100 から 100 までとすると y の値は最大が $20,000$ 、最小が 0 の曲線となります.



- ② この曲線を 640×400 ドットの画面に表示することを考えます.
- ③ まず x 方向の -100 から 100 は画面上では $+200$ すなわち、 100 ドットから 300 ドットとします.
- ④ y 方向の最大 $20,000$ は 200 ドットに縮小します.
- ⑤ y 方向の 0 の位置は画面の 300 ドット目とします. なお、画面は xy 方向は左上が $(0,0)$ で右下が $(639,399)$ ですから y 方向は普通の座標系とは方向が逆です. 画面は y のプラスが下、 y のマイナスが上となっています.
- ⑥ したがって、 $y=2x^2$ は画面上では次のようになります.



- ⑦ 実際の表示をします.

$y=2x^2$ を $\text{fna}(x)=2*x*x$ と定義します.

i を -100 から 99 までくり返し

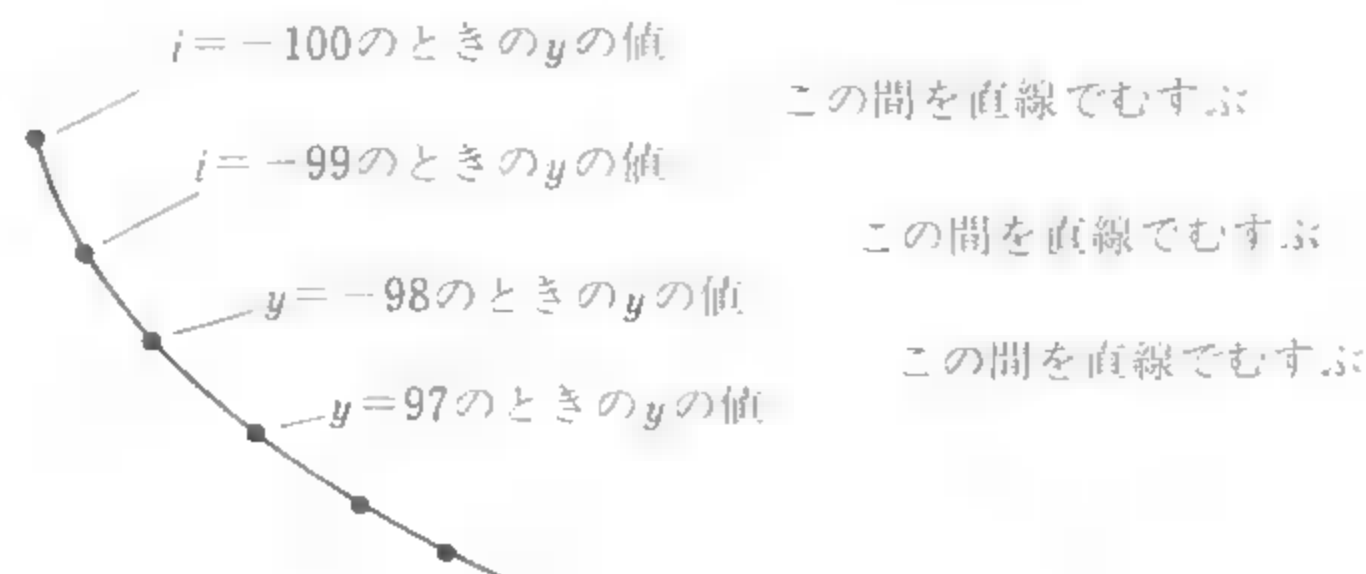
x ドットを $i+100$ とします.

y ドットを $300-\text{fna}(i)/100$ これが x が i のときの y の値です.

x ドットを $(i+1)+100$ とします.

y ドットを $300-fna(i+1)/100$ とします. これが x が $i+1$ のときの y の値です.

この 2 点をむすびます. このくり返しで曲線をかきます.



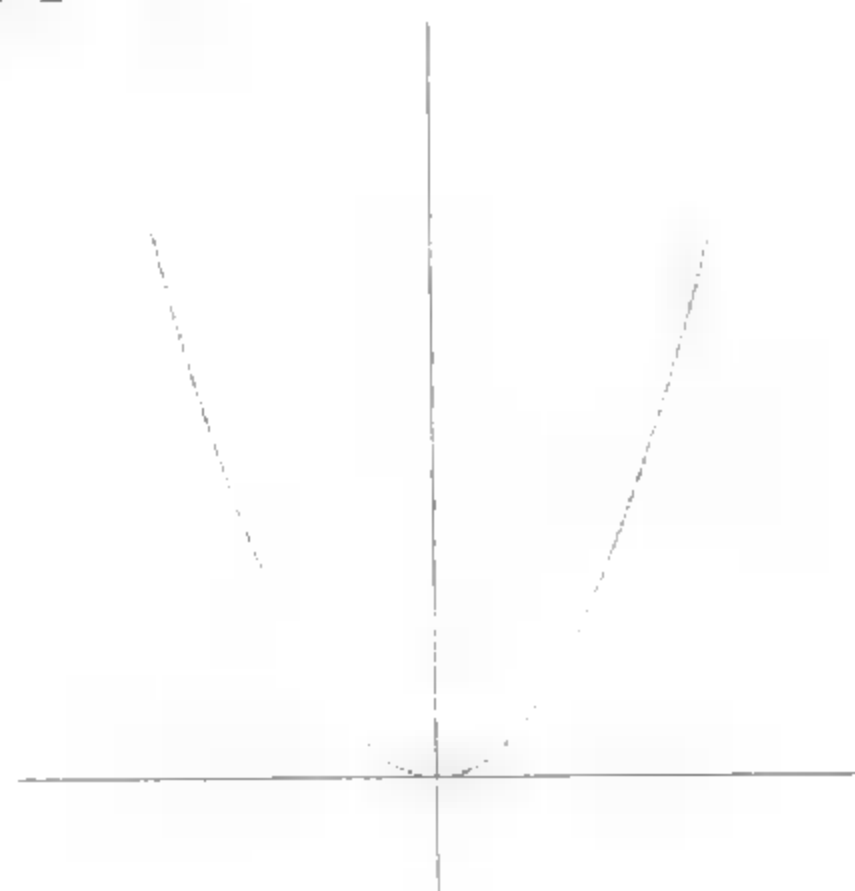
- ⑧ 表示の仕方としては, x を -100 としたときの y の値の位置にグラフィックポインタを移し, 次に, x を $-99, -98, \dots, 0, \dots, 100$ と変化させて y の値を求めその位置を次々に結ぶ方法もあります.

```
PSET(100,300-fna(-100)/100)
FOR i=-100 TO 100
  x=i+200
  y=300-fna(i)/100
  LINE-(x,y)
NEXT i
```

【プログラム例】

```
SCREEN 0
CLS 0
DEF fna (x) = 2 * x * x
LINE (50, 300)-(350, 300)
LINE (200, 10)-(200, 350)
FOR i = -100 TO 99
  y = 300 - fna(i) / 100
  yy = 300 - fna(i + 1) / 100
  LINE (i + 200, y)-(i + 201, yy)
NEXT i
END
```

【結 果】



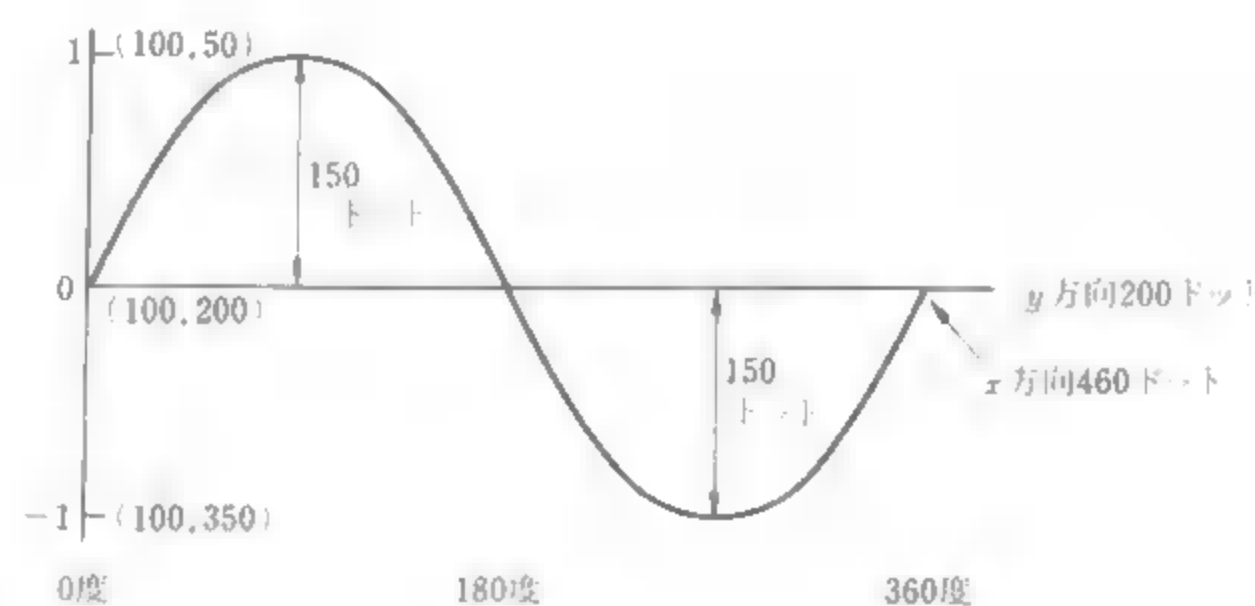
【例題 14】 関数式の図

SIN(x)のカーブを表示するプログラムをつくれ。

【解 説】 ◎ SIN(x)の式の図をかきます。

- ① SIN(x)は最大値が1, 最小値が-1です。0のときのy座標値を200として、1のとき50, -1のとき350すなわち1の幅を150ドットとなるようにします。その場合の定義は $fny(x)=200-150\text{SIN}(x)$ となります。

xは0度から360度までとし、0度するときx方向100ドット、360度するとき460ドットとします。図は5度おきにSIN(x)の値を求めて前の値とつないでいきます。

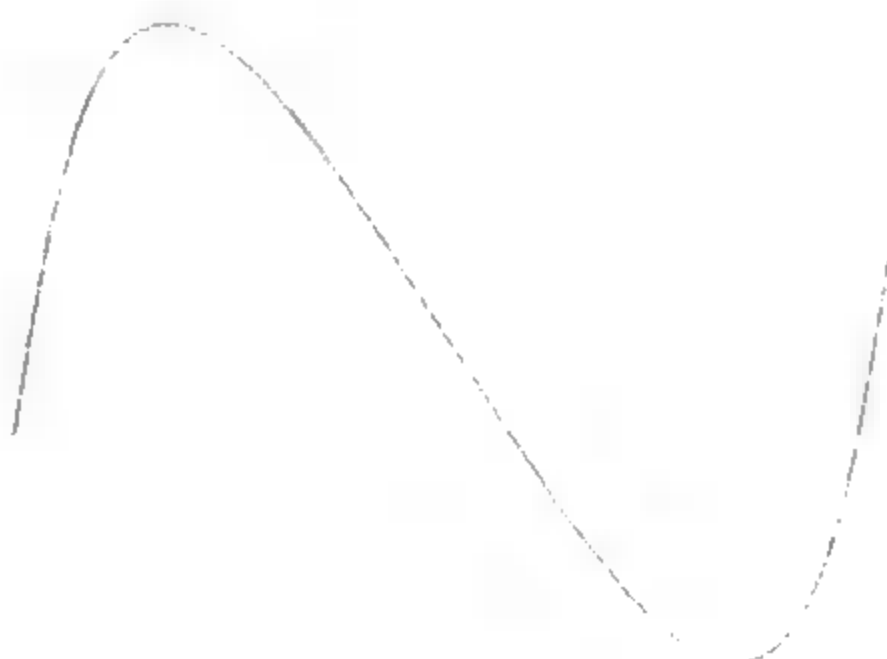


- ② SIN(x)でxが0度ときの値は(100, fny(0))です。ここにまず点をかき、次は(xx, yy)を求めて順にむすびます。xxは0度, 5度, 10度…360度のx方向の値で100, 105, 110……460ドットです。yyはsin(0度), sin(5度), sin(10度)……SIN(360度)のときのyドットです。fny(x度)となります。

【プログラム例】

```
SCREEN 0
CLS 0
DEF fny (x) = 200 - 150 * SIN(x)—— SIN(x)のy方向のスクリーン上のドット位置
PSET (100, fny(0))—— SIN(0)のときのドット位置をオンとする
FOR x = 0 TO 360 STEP 5 —— SIN(0度), SIN(5度)……SIN(360度)
    xx = x + 100
    r = x / 180 * 3.14159
    yy = fny(r)
    LINE -(xx, yy)—— のときのx, y座標値を求め順にむすぶ
NEXT x
END
```

【結 果】



〔演習問題〕

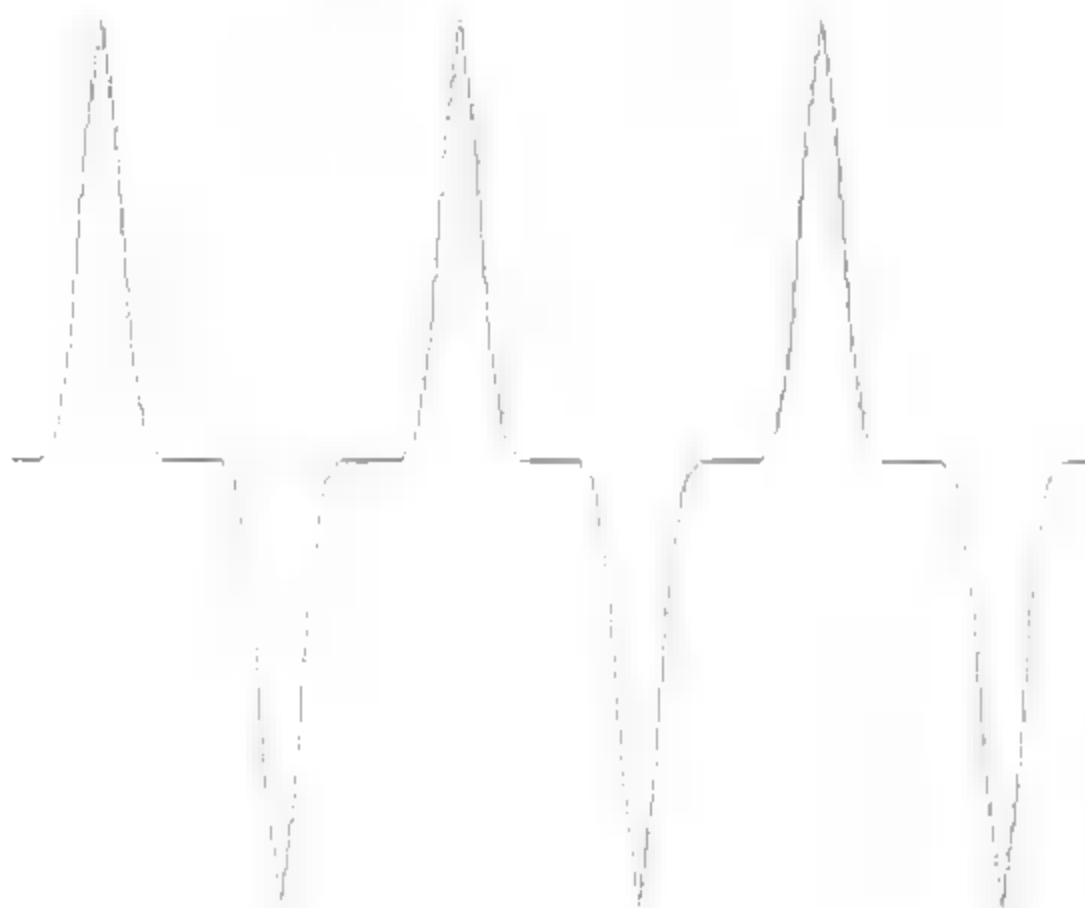
- (12) $y = \sin^9(3x)$ のグラフをかくプログラムをつくれ.
 (13) $y = 100\sin^3(8x) + 50\cos(3x)$ のグラフをかくプログラムをつくれ.

〔解 答〕

(12) 【プログラム例】

```
SCREEN 0
CLS 0
DEF fny (x) = 200 - 150 * SIN(3 * x) ^ 9 ——— SIN9(3x)の値の画面上の y 座標値を求める式の定義
PSET (100, fny(0)) ——— SIN(0)のときの x, y 座標値のドットをかく
FOR x = 0 TO 360 STEP 5 ——— SIN9(0)~SIN9(3*360)の値を5度おきに求め、
  xx = x + 100 ——— そのときの x, y 座標値を順にむすぶ
  r = x / 180 * 3.14159
  yy = fny(r)
  LINE -(xx, yy)
NEXT x
END
```

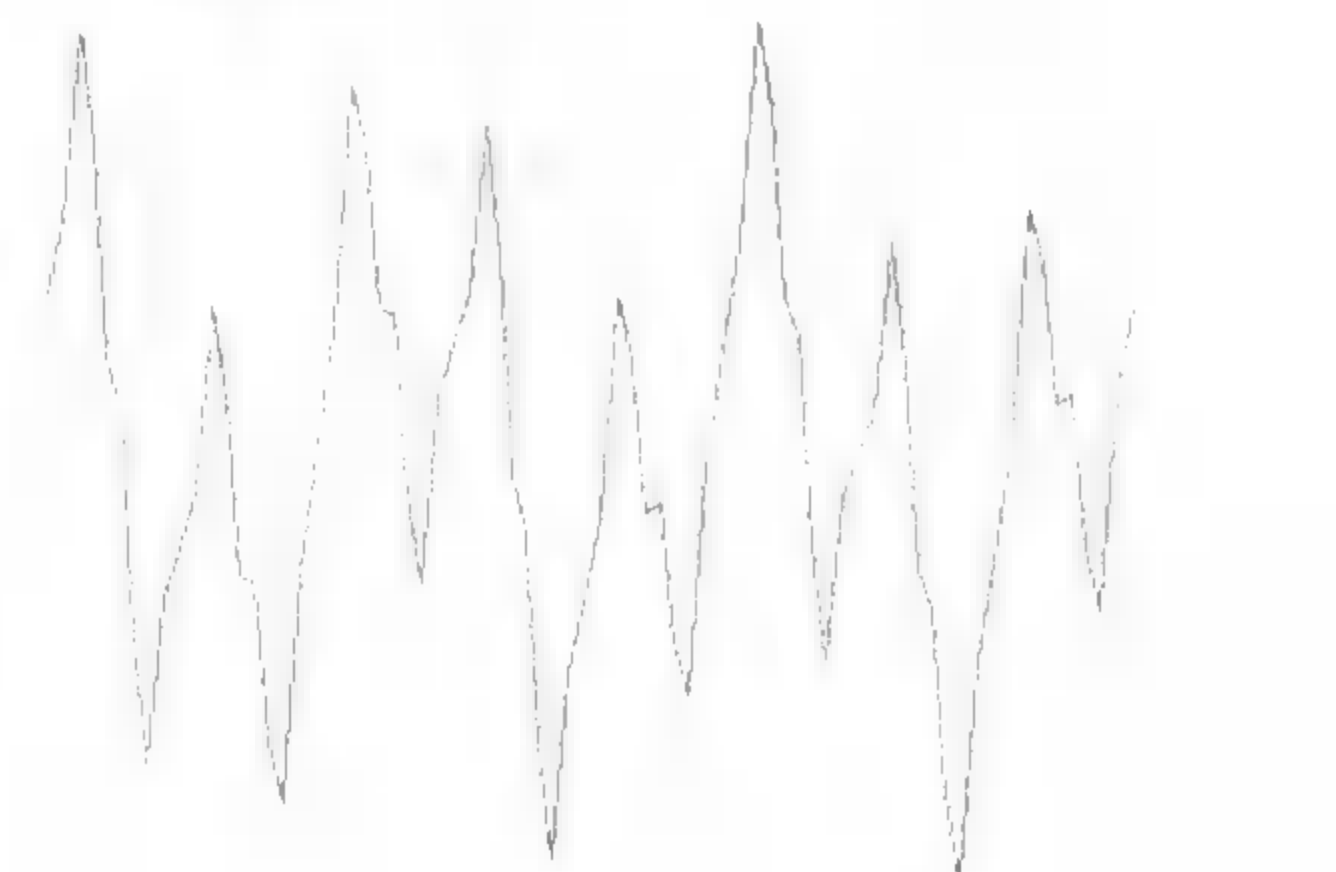
【結 果】



(13) 【プログラム例】

```
SCREEN 0
CLS 0
DEF fny (x) = 200 - 100 * SIN(8 * x) ^ 3 - 50 * COS(3 * x) ——— 式の定義
PSET (100, fny(0)) ——— 0度ときの fny の値の x, y 座標値のドットをかく
FOR x = 0 TO 360 STEP 5 ——— 0度~360度まで5度おきに式の値を求め、x, y 座標値を順にむすぶ
  xx = x + 100
  r = x / 180 * 3.14159
  yy = fny(r)
  LINE -(xx, yy)
NEXT x
END
```

【結 果】



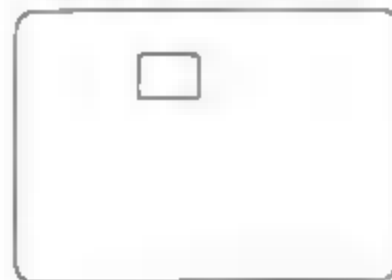
4章 画面切りかえウィンドウ・ビュー

SCREEN, WINDOW, WINDOW SCREEN, VIEW

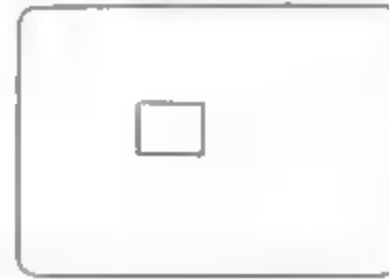
【例題 15】 画面の切りかえ

画面 0 に四角形、画面 1 に四角形をかき、画面 0 と 1 を切りかえて 10 回表示するプログラムをつくれ。

画面 0 の四角形



画面 1 の四角形



【解 説】 ◎ 画面切りかえ

① Quick BASIC では次のように書き込み画面と表示画面をもっています。

モード	ページ
84	0, 1, 2, 3
87	0 と 1
88, 0	0 と 1

② したがって、SCREEN0 の場合は書き込み、表示おのこの 2 画面ずつもっています。書き込み 2 画面は別々の図をかき、次に表示画面を切りかえると瞬時に図が入れかわります。アニメではよく使う手法です。

③ 画面 0 と画面 1 の 2 画面をもち、おのこの、独立して、書き込み、表示ができます。

0 画面への書き込み
↓
SCREEN 0,,0,0
↑
0 画面の表示

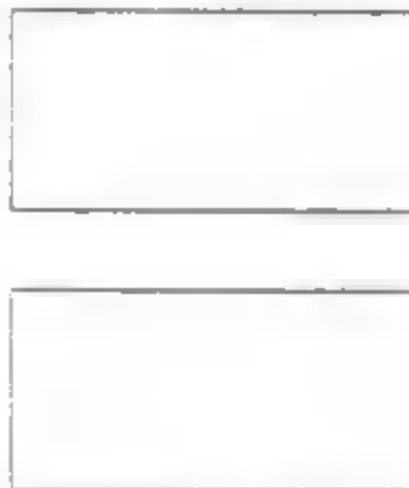
SCREEN 0,,1,1 ← 1 画面の表示
↑
1 画面への書き込み

④ 0 画面と 1 画面におのこの図を書き込んでおいて、表示画面を切りかえることで図が瞬時に入れかわります。

【プログラム例】

```
SCREEN 0, , 0, 0
CLS 0
LINE (200, 40)-(300, 90), , B
SCREEN , , 1, 1
LINE (200, 110)-(300, 160), , B
FOR i = 1 TO 10
  SCREEN , , , 0
  FOR j = 1 TO 1000: NEXT j
  SCREEN , , , 1
  FOR j = 1 TO 1000: NEXT j
NEXT i
END
```

【結果】



(上と下の四角が交互に表示される)

【例題 16】 ウィンドウ

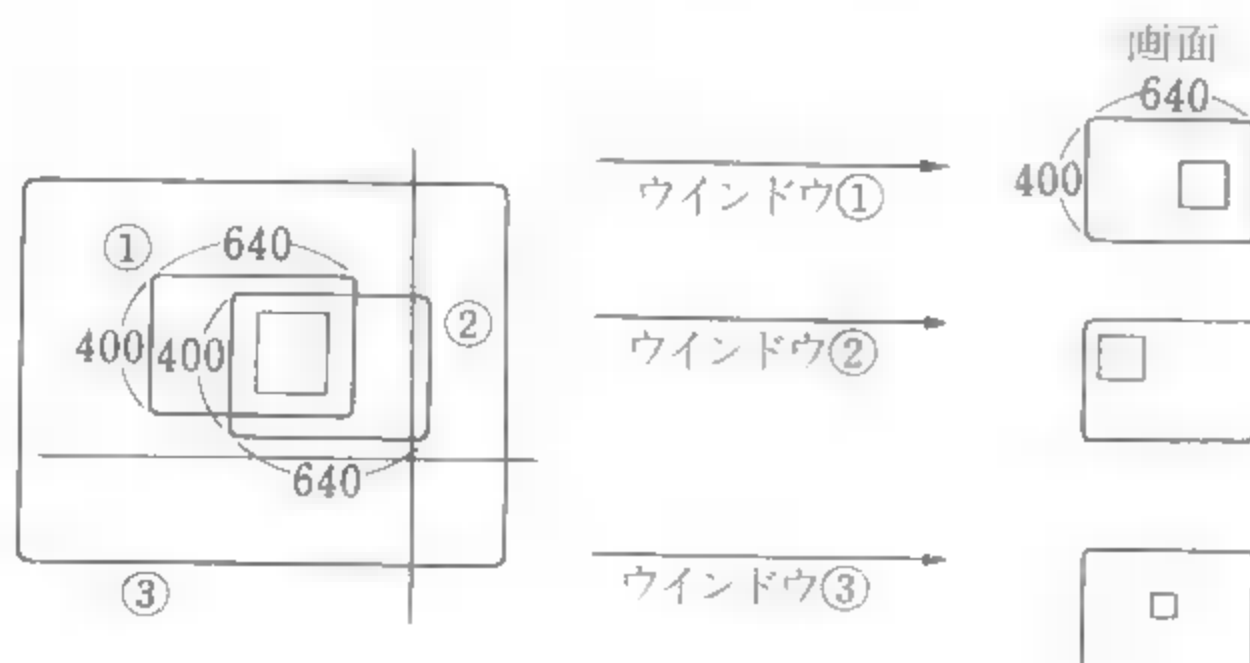
(-100, -200) - (200, 500) をウィンドウとして, (-50, 50) と (100, 150) を結ぶ直線と, ウィンドウを (-500, -1500) - (1000, 1000) として (-100, -200) と (200, 500) を結ぶ直線にかくプログラムをつくれ.

【解説】 ㊦ ウィンドウの使い方を示します.

- ① ワールド座標系の中でかいた図をあるウィンドウを設定してその部分を画面へ表示します. これはちょうど, 外の景色を窓を通して見るようなものです. 窓の大きさ, 位置によって表示される図の位置, 大きさがかわります.

ウィンドウの例①

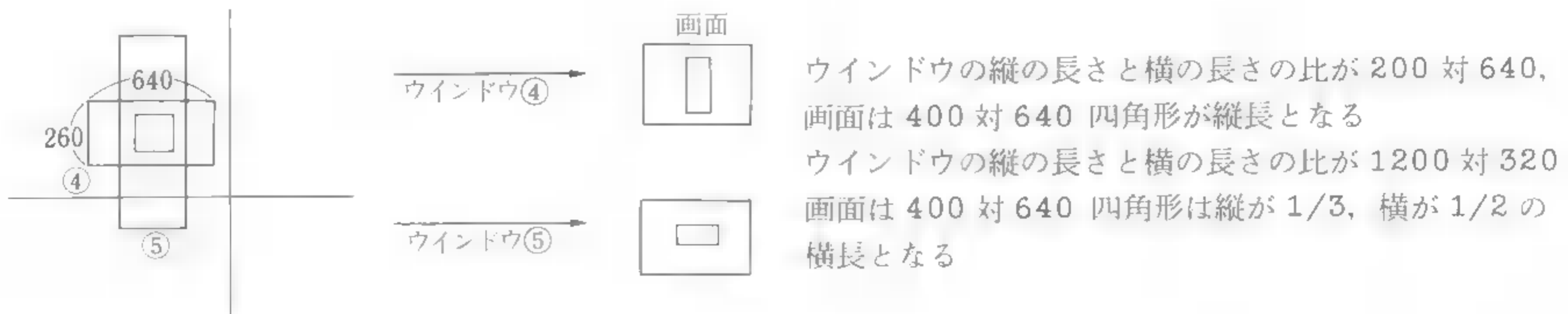
ワールド座標系の中の図 (正方形)



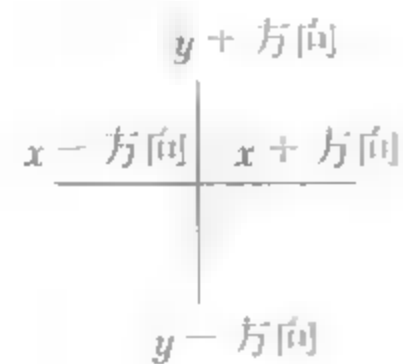
ワールド座標系の中でのウィンドウの大きさ = 画面の大きさ
正方形は右側
同上, ただし, ウィンドウの位置が異なる
正方形は左上
ウィンドウの大きさが画面の大きさの 3 倍 (縦, 横とも)
正方形は画面中央で 1/3 (縦, 横とも)

ウインドウの例②

ワールド座標系の中の図（正方形）

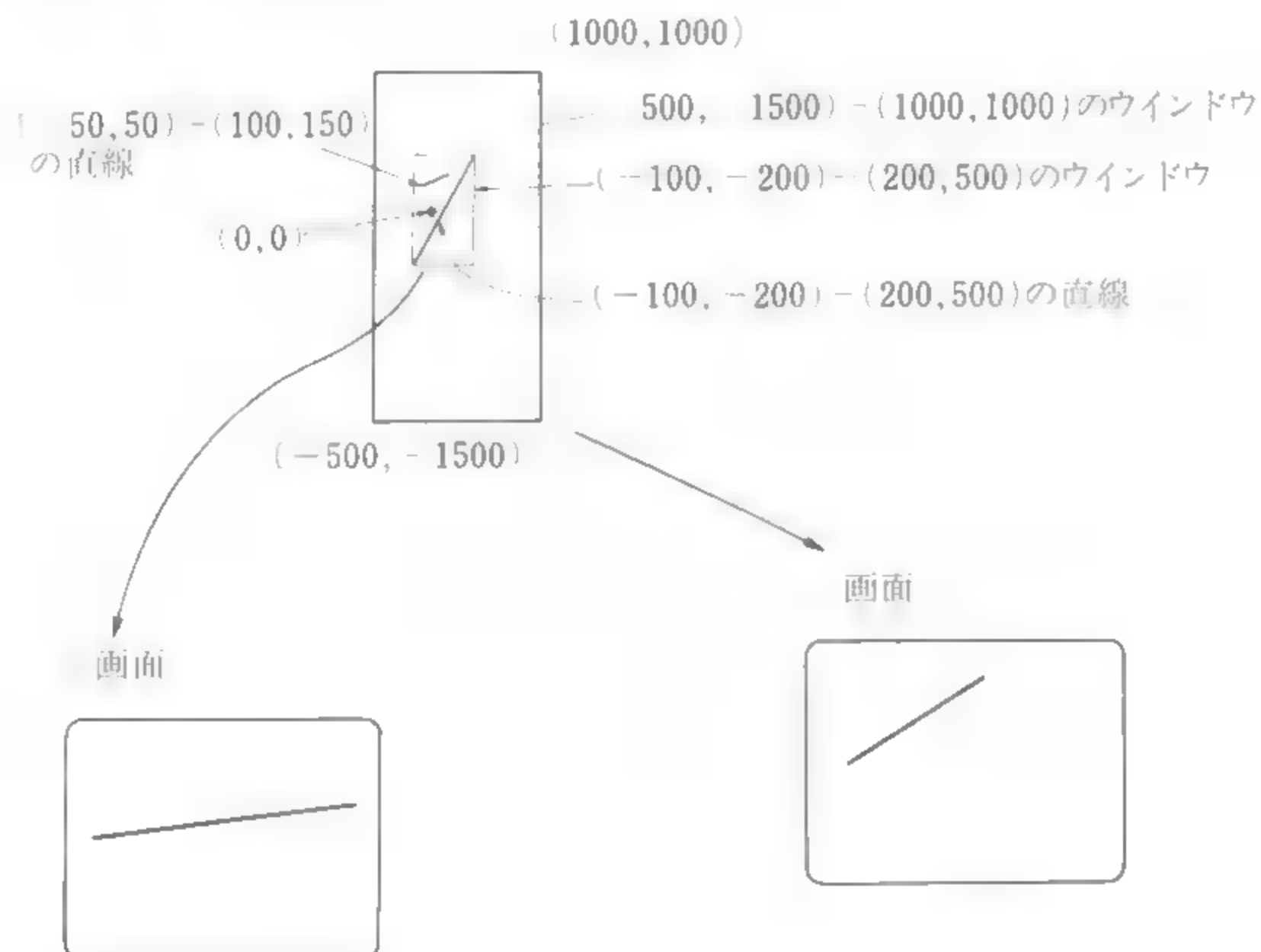


② WINDOW の座標系は次のとおりです。



③ 項①の図でみたように、切りとったウインドウ全体が画面全体となるように調整されて表示されます。

④ 例題の図は次のようになります。



【プログラム例】

```

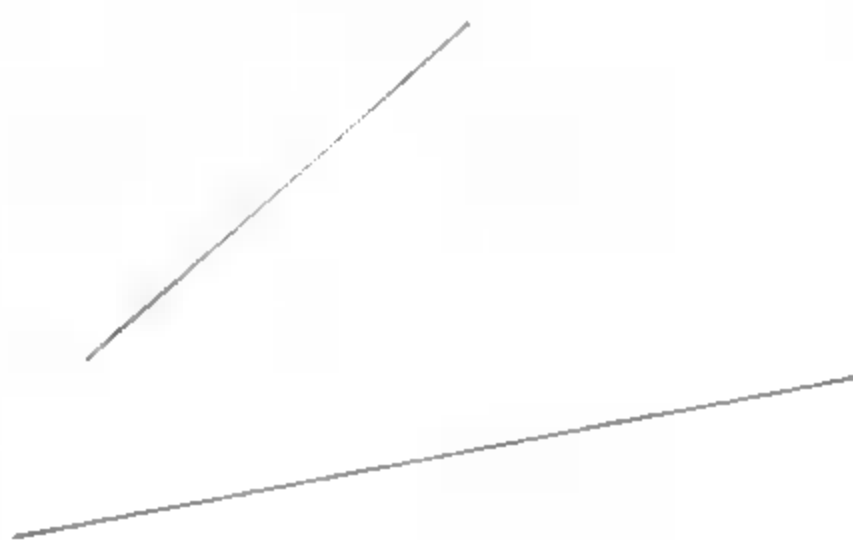
SCREEN 0, , 0, 0
CLS 0
WINDOW (-100, -200)-(200, 500)
LINE (-50, 50)-(100, 150)
WINDOW (-500, -1500)-(1000, 1000)
LINE (-100, -200)-(200, 500)
END

```

ウインドウの設定と直線

ウインドウの設定と直線

【結 果】

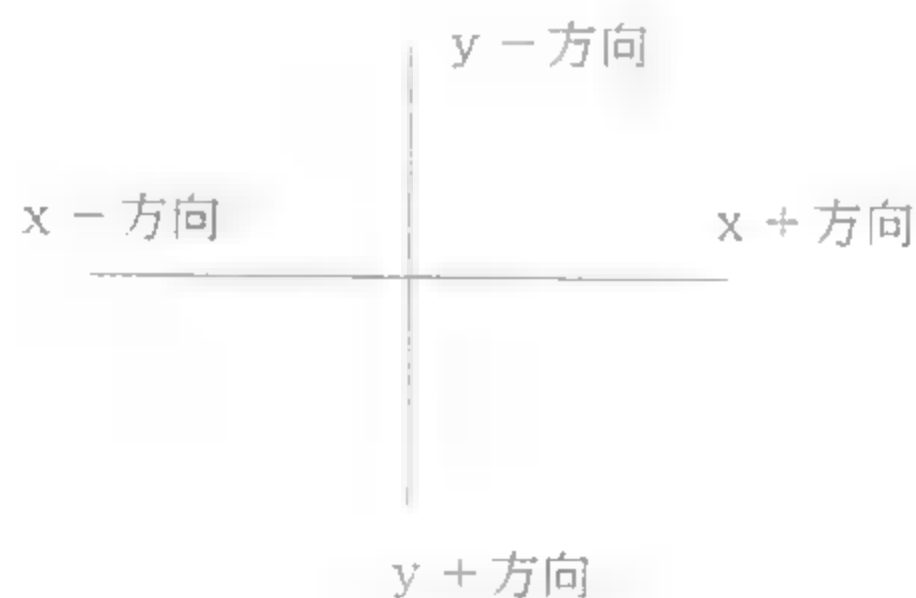


【例題 17】 ウィンドウスクリーン

【例題 16】の図を逆にして表示するプログラムをつくれ。

【解 説】 ㉟ ウィンドウスクリーンを設定して図を表示します。

- ① WINDOW SCREEN は次のような座標系を考えます。WINDOW と y 方向のプラスとマイナスが逆になるほかは全く同じです。



- ② したがって、WINDOW と WINDOW SCREEN を使い同じ値で設定して同じ図をかくと縦方向が逆となります。

【プログラム例】

```
SCREEN 0, , 0, 0
CLS 0
WINDOW SCREEN (-100, -200)-(200, 500)——— ウィンドウの設定と図
LINE (-50, 50)-(100, 150) —————
WINDOW SCREEN (-500, -1500)-(1000, 1000)——— ウィンドウの設定と図
LINE (-100, -200)-(200, 500) —————
END
```

【結 果】



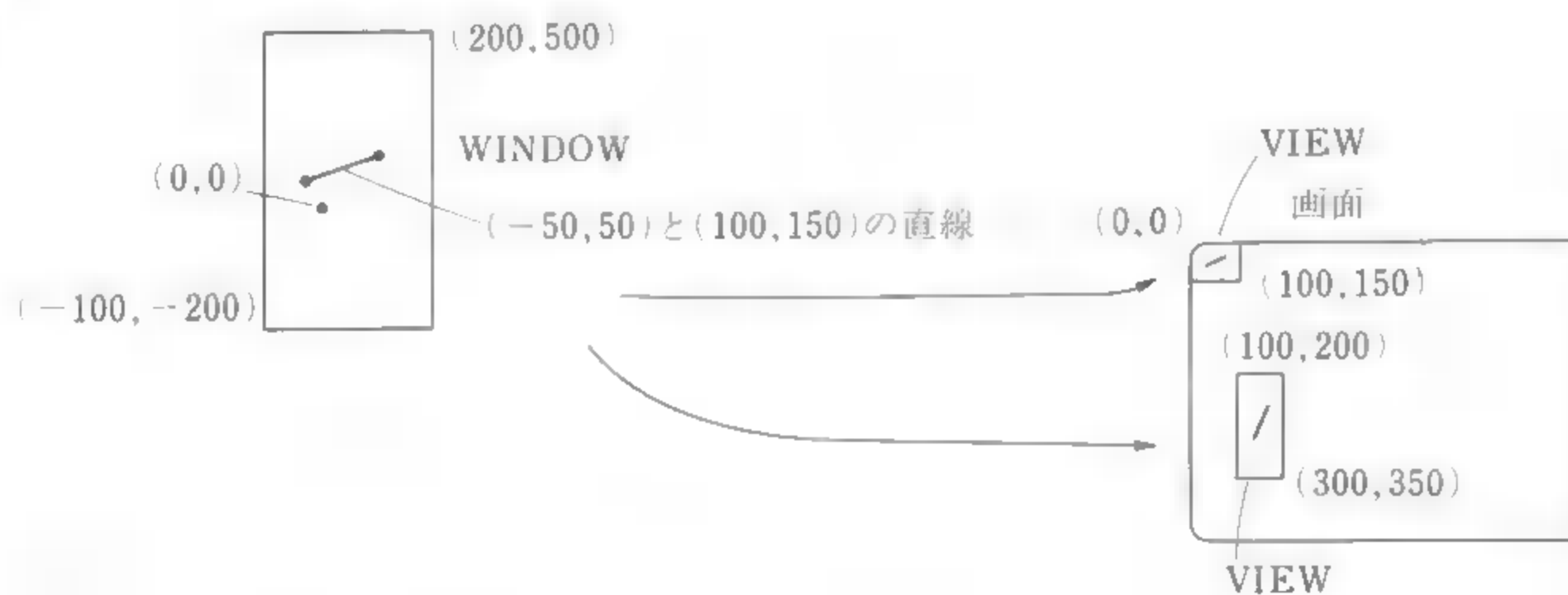
【例題 18】 ビュー

(-50, 50)と(100, 150)を結ぶ直線を(-100, -200)と(200, 500)を対角とするウインドウで切って(0, 0)と(100, 50)を対角とするビューに表示し、次に同じ直線を(100, 200)と(300, 350)を対角とするビューに表示するプログラムをつくれ。

【解 説】 ㊦ ビューを指定します。

① VIEW は画面上の対角を示して、その範囲にウインドウの図を表示します。したがって、ウインドウで切りとった図を画面の任意の位置に表示できます。

② WINDOW, VIEW, LINE は次のようになります。



③ VIEW の設定のない場合は画面全体がビューです。

【プログラム例】

```
SCREEN 0, , 0, 0
CLS 0
WINDOW (-100, -200)-(200, 500)———ウインドウの設定
VIEW (0, 0)-(100, 50)———ビューの設定
LINE (-50, 50)-(100, 150)———直線
VIEW (100, 200)-(300, 350)———ビューの設定
LINE (-50, 50)-(100, 150)———直線
END
```

【結 果】



〔演習問題〕

(14) 画面 0 と画面 1 に、左矢印と右矢印をかき、画面の切りかえで交互に 10 回表示するプログラムをつくれ。

- (15) 画面 0 に四角形、画面 1 にその四角形と同じ四角形の中に円をかいた図をつくり、交互に 10 回表示するプログラムをつくれ。
- (16) $(-1000, -1000) - (1000, 1000)$ をウインドウとして $(-500, 500)$ と $(100, 150)$ を対角とする箱、ウインドウを $(-500, -500) - (1000, 1000)$ として、 $(-500, 500)$ を $(100, 150)$ を対角とする箱、ウインドウを $(-1000, -1000) - (500, 5000)$ として $(-500, 500)$ と $(100, 150)$ を対角とする箱をかくプログラムをつくれ。
- (17) $(-100, -200) - (200, 500)$ をウインドウとし、 $(-50, 50)$ と $(100, 150)$ を対角とする箱をビューを $(0, 0) - (100, 100)$, $(100, 100) - (200, 200)$, $(0, 0) - (639, 399)$ の 3 種にして表示するプログラムをつくれ。

〔解 答〕

(14) 【プログラム例】

```
SCREEN 0, , 0, 0 ——— 左矢印を画面 0 にかいて表示
CLS 0
LINE (200, 40)-(300, 40)
LINE (220, 35)-(200, 40)
LINE -(220, 45)
SCREEN , , 1, 1 ——— 右矢印を画面 1 にかいて表示
LINE (200, 40)-(300, 40)
LINE (280, 35)-(300, 40)
LINE -(280, 45)
FOR i = 1 TO 10 ——— 画面 0 と 1 を交互に 10 回表示
    SCREEN , , , 0
    FOR j = 1 TO 1000: NEXT j
    SCREEN , , , 1
    FOR j = 1 TO 1000: NEXT j
NEXT i
END
```

【結 果】



(矢印が同じ場所で交互に表示)



(15) 【プログラム例】

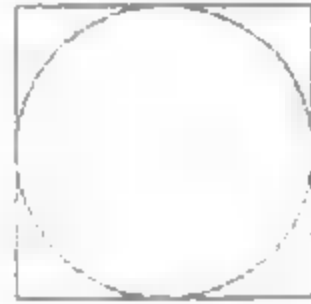
```
SCREEN 0, , 0, 0 ——— 画面 0 に四角形をかいて表示
CLS 0
LINE (100, 50)-(200, 150), , B ———
```

```

SCREEN , , 1, 1 ———— 画面 1 に四角形と円をかいて表示
LINE (100, 50)-(200, 150), , B
CIRCLE (150, 100), 50
FOR i = 1 TO 10 ———— 交互に 10 回表示
    SCREEN , , , 0
    FOR j = 1 TO 1000: NEXT j
    SCREEN , , , 1
    FOR j = 1 TO 1000: NEXT j
NEXT i
END

```

【結 果】



左図と右図が同じ場所で交互に表示される

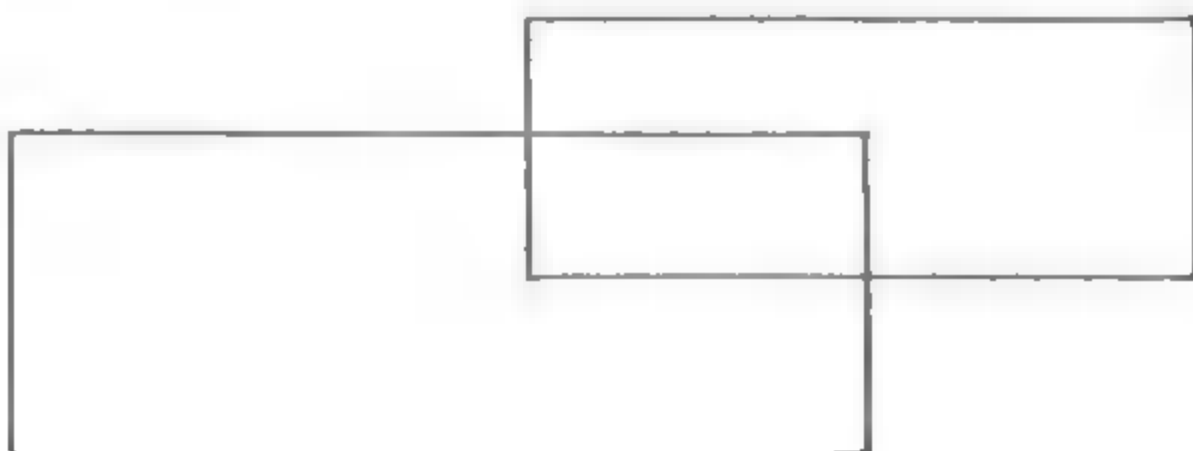
(16) 【プログラム例】

```

SCREEN 0, , 0, 0
CLS 0
WINDOW (-1000, -1000)-(1000, 1000) ———— ウィンドウの設定と箱の表示
LINE (-500, 500)-(100, 150), , B
WINDOW (-500, -500)-(1000, 1000) ———— ウィンドウの設定と箱の表示
LINE (-500, 500)-(100, 150), , B
WINDOW (-1000, -1000)-(500, 5000) ———— ウィンドウの設定と箱の表示
LINE (-500, 500)-(100, 150), , B
END

```

【結 果】



(17) 【プログラム例】

```

SCREEN 0, , 0, 0
CLS 0
WINDOW (-100, -200)-(200, 500) ———— ウィンドウの設定
VIEW (0, 0)-(100, 100) ———— ビューの設定
LINE (-50, 50)-(100, 150), , B ———— 四角形の表示
VIEW (100, 100)-(200, 200) ———— ビューの設定
LINE (-50, 50)-(100, 150), , B ———— 四角形の表示

```

```
VIEW (0, 0)-(639, 399) ———— ビューの設定  
LINE (-50, 50)-(100, 150), , B ——— 四角形の表示  
END
```

【結 果】



5章 カラー

PALETTE, COLOR, SCREEN

【例題 19】 カラー

四角形をかき、次に黒から黄まで 6 色でかきかえるプログラムをつくれ。

【解 説】 ㊦ カラーを変化させます。

- ① 4 行目で四角形をかきます。このとき、線の色は 4 によって 色でかきます。これは標準の指定で 3 行目の PALETTE を指定します。

0 黒	1 青	2 緑	3 水色
4 赤	5 紫	6 黄	7 白

- ② 6 行目の PALETTE 4, i によって、i が 0 から 6 まで変化しますので、4 が 0 の色、1 の色… …6 の色になります。したがって、四角形が黒、青、緑、 ……となります。なお、黒は背景が黒で色が黒ですから実際は見えません。

PALETTE x, y は値 x を色番号 y でわりあてます。

- ③ したがって、同じ赤でかく場合でも例えば次のようにできます。

```
PALETTE 4,4
```

```
LINE(100,100)-(200,150),4,B
```

```
PALETTE 2,4
```

```
LINE(100,100)-(200,150),2,B
```

```
PALETTE 5,4
```

```
LINE(100,100)-(200,150),5,B
```

【プログラム例】

```
SCREEN 0
```

```
CLS 0
```

```
PALETTE—————色コードを既定値にもどす
```

```
LINE (100, 100)-(200, 150), 4, B————4 の色（赤）で四角形をかく
```

```
FOR i = 0 TO 6 —————4 に相当する色を 0～6 の色に変える
```

```
    PALETTE 4, i
```

```
    FOR j = 1 TO 1000: NEXT j
```

```
NEXT i
```

```
PALETTE—————色コードを既定値にもどす
```

```
END
```


【結 果】



赤, 黒, 青, 緑, 水色, 赤, 紫, 黄, 赤で表示

【例題 20】 文字の色

"Quick Basic"を青, 赤, 水色でかくプログラムをつくれ.

【解 説】 ◎ 文字の色を指定します.

- ① a\$を"Quick Basic"とします.
- ② COLOR 1 として PRINT a\$を行うと 1 の色番号の色で a\$を表示します. すなわち青です. 色番号は次の色を示します.

0 黒 1 青 2 緑 3 水色 4 赤 5 紫 6 黄 7 白

8~15 は 0~7 の反転, 16~31 は 0~15 の点滅

【プログラム例】

```
SCREEN 0
CLS 0
a$ = "Quick Basic"
COLOR 1
PRINT a$
COLOR 4
PRINT a$
COLOR 3
PRINT a$
COLOR 7
END
```

【結 果】

```
Quick Basic (青)
Quick Basic (赤)
Quick Basic (水色)
```

〔演習問題〕

- (18) 結果の図をつくるプログラムをつくれ.
- (19) 四角形を 2 つ上下に赤色と緑色で塗り, 次に上の四角を赤と青と, 下の四角を緑と黄に切りかえることを 10 回くり返すプログラムをつくれ.

〔 解 答 〕

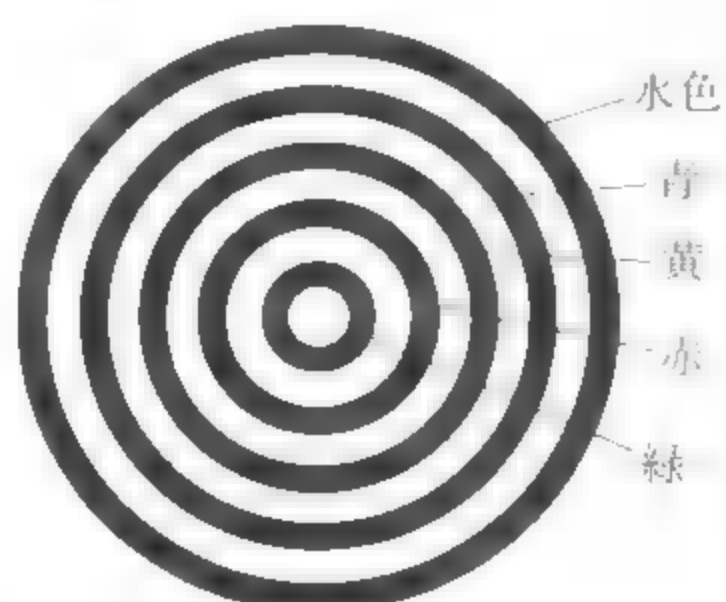
(18) 【プログラム例】

```

SCREEN 0
CLS 0
FOR i = 1 TO 10
  CIRCLE (300, 200), i * 10
  IF (i MOD 2) = 0 THEN PAINT (300 + i * 10 - 5, 200), i MOD 7, 7
NEXT i
END

```

【結 果】



(19) 【プログラム例】

```

SCREEN 0
CLS 0
PALETTE ————— 色コードを既定値にもどす
LINE (100, 100)-(200, 150), 4, BF ———— 四角を赤で塗る
LINE (100, 200)-(200, 250), 2, BF ———— 四角を緑で塗る
FOR i = 1 TO 10 ————— 2つの四角形の色を切りかえる
  PALETTE 4, 4
  PALETTE 2, 2
  FOR j = 1 TO 200: NEXT j
  PALETTE 4, 1
  PALETTE 2, 6
  FOR j = 1 TO 200: NEXT j
NEXT i —————
PALETTE ————— 色コードで既定値にもどす
END

```

【結 果】



赤と青の切りかえ



緑と黄の切りかえ

6 章 図形の配列へのとりこみと表示

GET, PUT

【例題 21】 図形の配列へのとり込みと表示

下図を表示して全体を配列 a%, 右上 1/4 を配列 b% にとり込んで、次に全体を、(200, 50) を左上として、1/4 の図を (350, 50) を左上として表示するプログラムをつくれ。



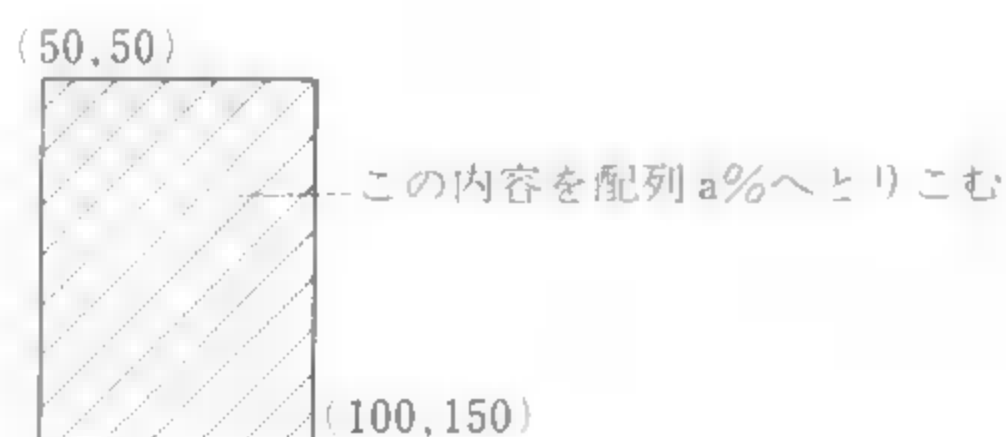
【解 説】 ◎ 図を配列へとり込み、任意の位置へ表示します。

- ① 4 行目の CIRCLE 文から 8 行目の LINE 文までで、問題の図をつくります。
- ② 次の形で図を配列へとり込みます。

GET(50,50)-(150,150),a%

左上と右下を示す座標値を対角
とする四角形の内容
配列にとりこむ内容

とり込む配列名
配列は前もって充分量を確認しておく



a% の内容

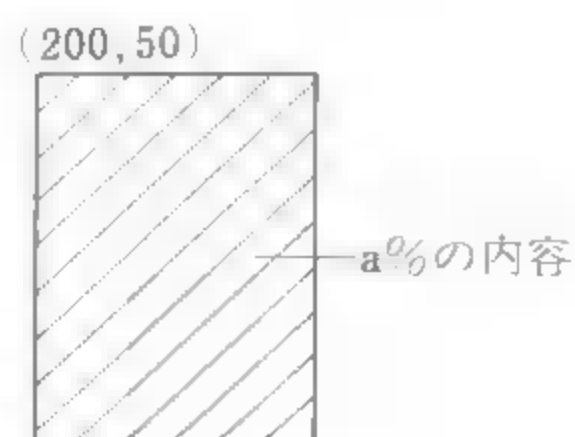


- ③ 次の形で任意の位置に表示します。

PUT(200,50),a%,PSET

表示をする左上
の座標

表示を示す
表示する配列



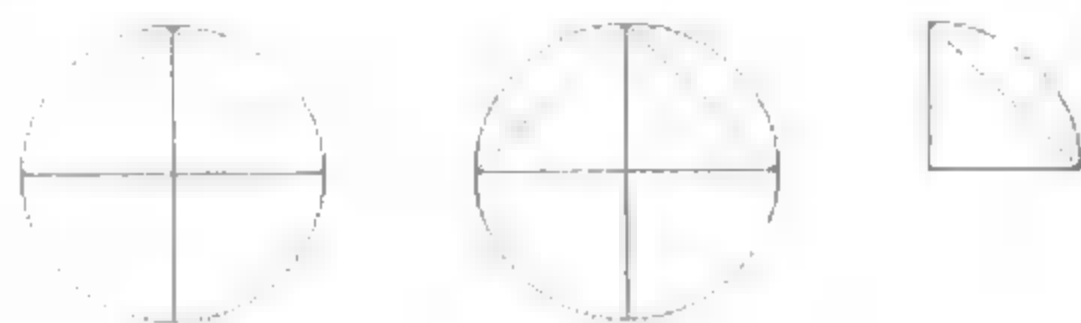
【プログラム例】

```

SCREEN 0
CLS 0
DIM a%(2700), b%(800)          図をとりこむ配列
CIRCLE (100, 100), 50          図の表示
LINE (100, 50)-(150, 100)
LINE -(50, 100)
LINE -(100, 50)
LINE -(100, 150)
GET (50, 50)-(150, 150), a%    図形のとり込み
GET (100, 50)-STEP(50, 50), b%
PUT (200, 50), a%, PSET        図形を表示
PUT (350, 50), b%, PSET
END

```

【結 果】



【例題 22】 オプション付きの表示

【例題 21】と同じ図形をかき全体を配列 a%, 右上 1/4 を配列 b%にとりこんで、次に (90, 190)-(160, 260)と(190, 190)-(260, 260)の四角形を白でうめておいてから、①a%の内容を左上を(200, 50)として PRESET 付き、②b%の内容を左上を(50, 50)として OR 付き、③b%の内容を左上を(100, 200)として XOR 付き、④b%の内容を左上を(200, 200)として AND 付きで表示するプログラムをつくれ。

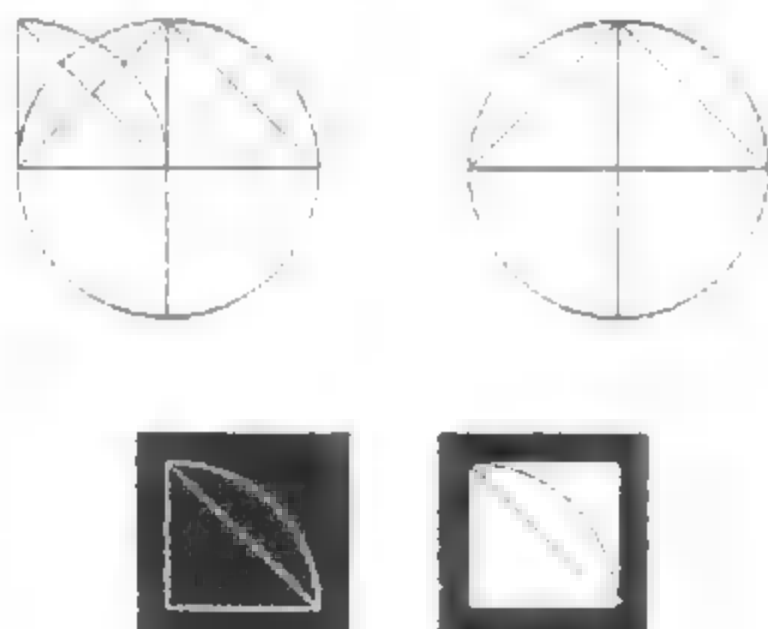
【解 説】 ◎ PUT にオプションをつけて表示します。

- ① PUT は PSET の他、PRESET, OR, XOR, AND のオプションがつけられます。
- ② 10 行目までは【例題 21】と同じで図を配列 a%と b%にとり込みます。
- ③ 11~12 行目は四角形を 2 つ塗りつぶします。
- ④ PUT で図を表示する場合、PRESET を指定すると、すでに表示がしてある場合は PUT する内容と同じドットは消去、表示のない場合は無表示です。
- ⑤ PUT で表示する場合 OR をつけると、表示するドットと、現在の画面のドットの状態の OR をとってドットの状態を決めます。
- ⑥ PUT で表示する場合 XOR をつけると表示するドットと、現在の画面のドットの状態の XOR をとってドットの状態を決めます。
- ⑦ PUT で表示する場合 AND をつけると表示するドットと、現在の画面のドットの状態の AND をとってドットの状態を決めます。

【プログラム例】

```
SCREEN 0
CLS 0
DIM a%(2700), b%(800)
CIRCLE (100, 100), 50
LINE (100, 50)-(150, 100)
LINE -(50, 100)
LINE -(100, 50)
LINE -(100, 150)
GET (50, 50)-(150, 150), a%
GET (100, 50)-STEP(50, 50), b%
LINE (90, 190)-(160, 260), , BF
LINE (190, 190)-(260, 260), , BF
PUT (200, 50), a%, PRESET
PUT (50, 50), b%, OR
PUT (100, 200), b%, XOR
PUT (200, 200), b%, AND
END
```

【結 果】



〔演習問題〕

- (20) 結果図の下側の背景の中で塗りつぶした円を左から右へ動かすプログラムをつくれ。
- (21) 結果の図をつくり、配列 a%にとりこんで、次に画面左から右へ3つ表示するプログラムをつくれ。
- (22) 結果の図を左から右へ動かすプログラムをつくれ。

〔解 答〕

(20) 【プログラム例】

```
SCREEN 0
CLS 0
DIM a%(500) ————— 図をとりにこむ配列
CIRCLE (50, 200), 20 ————— 図の表示
PAINT (50, 200), 7, 7 —————
```

```

GET (30, 180)-(70, 220), a%——図形のとり込み
CLS 0
LINE (30, 250)-(620, 250)——バックの表示
LINE -(620, 180)
LINE -(550, 210)
LINE -(400, 205)
LINE -(300, 215)
LINE -(100, 220)
LINE -(30, 210)
LINE -(30, 250)
PAINT (50, 230), 7, 7
FOR i = 50 TO 580 STEP 40——円を左から右へ移動
    PUT (i, 200), a%, OR
FOR j = 1 TO 500: NEXT j
NEXT i
END

```

【結 果】



(21) 【プログラム例】

```

SCREEN 0
CLS 0
DIM a%(2700)
CIRCLE (100, 100), 50——図形の表示
CIRCLE (115, 80), 10, , , .5
CIRCLE (85, 80), 10, , , .5
LINE (103, 65)-(127, 65)
LINE (73, 65)-(97, 65)
LINE (100, 100)-(100, 130)
LINE (80, 140)-(120, 140)
GET (50, 50)-(150, 150), a%——図形のとりこみ
FOR i = 1 TO 3——図形の表示
    PUT (150 * i, 200), a%, PSET
NEXT i
END

```

【結 果】



(22) 【プログラム例】

```

SCREEN 0
CLS 0
DIM a%(850)
LINE (100, 100)-(160, 130), , B
LINE (130, 90)-(140, 95)
LINE -(130, 100)
LINE -(120, 95)
LINE -(130, 90)
LINE (110, 105)-(120, 115), , B
LINE (140, 105)-(150, 125), , B
CIRCLE (115, 135), 5
CIRCLE (145, 135), 5
GET (100, 90)-(160, 140), a%
CLS 0
FOR i = 1 TO 50
    PUT (i * 10, 200), a%, PSET
    PUT (i * 10, 200), a%, XOR
NEXT i
END

```

図形の表示

図形のとり込み

図形の表示・消去

【結果】



(左から右へ動く)

7 章 POINT と DRAW

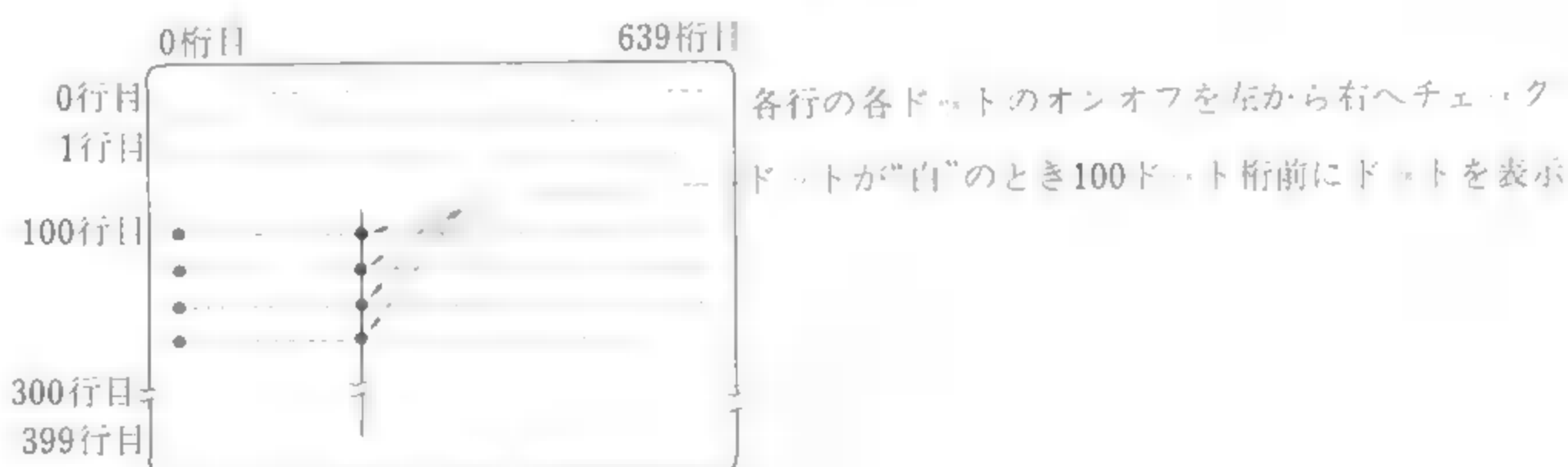
POINT, DRAW

【例題 23】 指定位置の座標値

(100,100)と(100,300)を直線で結び、次に画面を1ドットずつスキャンして、ドットがオンのときそのドット位置の左100ドットの位置のドットをオンにするプログラムをつくれ。

【解 説】 ① グラフィックカーソルのある位置の座標値

- ① `LINE(100,100)-(100,300)`で縦線をかきます。
- ② 左上(0,0)から(639,0)まで一番上のドット行を左から右へスキャンしていった、ドットがオンのとき `x` 方向に-100 の位置のドットをオンします。
- ③ 同様に 1 行~399 行まで同じことをくり返します。



【プログラム例】

```
SCREEN 0
CLS 0
LINE (100, 100)-(100, 300)
FOR i = 0 TO 399
  FOR j = 0 TO 639
    IF POINT(j, i) = 7 THEN PSET (j - 100, i)
  NEXT j
NEXT i
END
```

縦線の表示

0 行から 399 行まで 0 桁から 639 桁までドットが白かどうかチェック。白のとき 100 ドット桁前の 1 ドットを表示

【結 果】

【例題 24】 DRAW を使った図

DRAW を使って、結果の図をつくれ。

【解 説】 ◎ DRAW を使って図をかきます。

- ① DRAW の後に "アルファベット+数値" をおいてグラフィックマクロ言語をつくります。
- ② 機能は次のとおりです。例題を通して順に説明していきます。

① カーソル移動

B, N

U, D, L, R, E, F, G, H, M

② 回転, 色指定, 拡大, 縮小

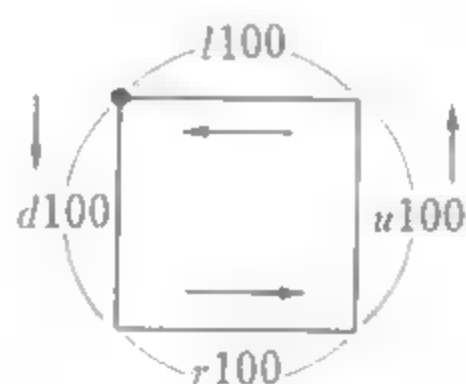
A, TA, C, S, P

③ 部分文字列

- ③ 3 行目の S2 は拡大率が 2 です。S は拡大縮小を示し、2 は倍率です。2 の部分は 1 から 255 が使えます。実際の値はカーソル移動命令で与えられる数値にこの値を掛けたものです。

たとえば U50 は上へ 50 ドット移動しますが、S2 が指定してあると上へ $50 \times 2 = 100$ ドット移動します。

- ④ 4 行目の d100, r100, u100, l100 はおののおの、下、右、上、左へ 100 ドットの移動を示します。ただし、S2 が指定してありますから本例ではその 2 倍の量の移動です。



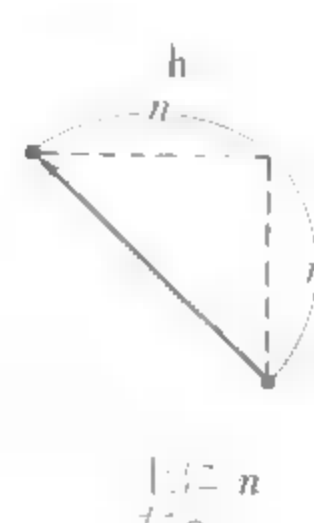
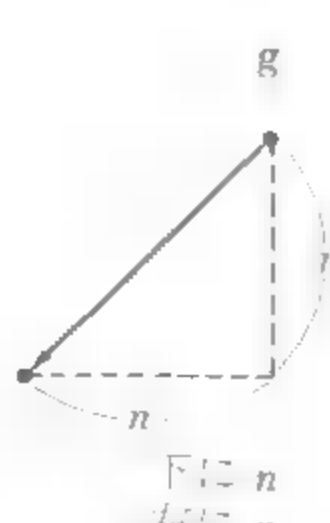
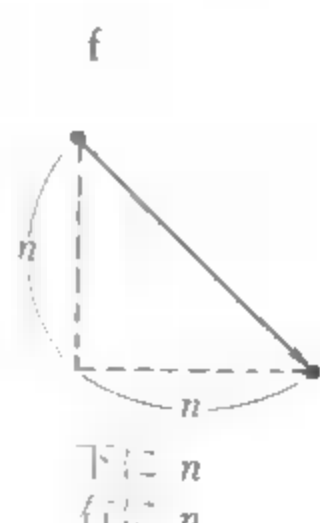
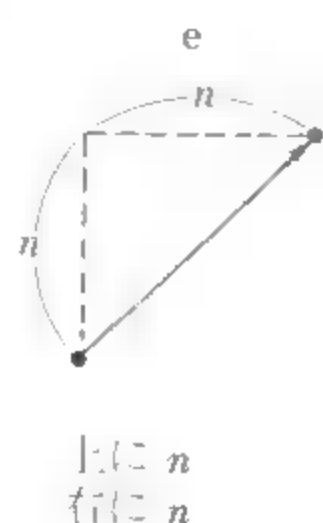
- ⑤ 5 行目の bm 100, 100 は次の意味です。

B が線をひかずにカーソルのみ移動させます。

Mx, y は座標値 (x, y) まで移動します。すなわち、BM100, 100 はカーソルのみ (100, 100) の位置へ移します。

BM+x, +y のように、x, y に符合をつけると現在の位置から相対位置を示します。プラスまたはマイナス x, プラスまたはマイナス y だけ移した位置を示します。

- ⑥ 7 行目の e30 f30 d60 g30 h30 u60 の e, f, g, h はおののおの次のとおりです。



- ⑦ 8 行目の nm+100, +0 の n は移動位置まで線をひいた後 (0, 0) までカーソルをもどします。し

たがって、nm+100, +0 は現在位置から x 方向 100, y 方向 0 の位置までの横線をひいた後、カーソルを(0,0)へもどします。

【プログラム例】

```
SCREEN 0
CLS 0
DRAW "s2"————— 2 倍の倍率
DRAW "d100 r100 u100 l100"————— 四角をかく
DRAW "bm100,100"————— カーソルを移す
DRAW "s5"————— 5 倍の倍率
DRAW "e30 f30 d60 g30 h30 u60"————— 6 角形の表示
DRAW "bm0,0 nm+100,+0"————— (0,0)から横線をひいてカーソルを(0,0)へもどす
END
```

【結 果】



【例題 25】 DRAW を使った図

四角形をかき、次にそれを塗りつぶし、次に四角形を 0 度、90 度、180 度、270 度回転した図をかくプログラムをつくれ。

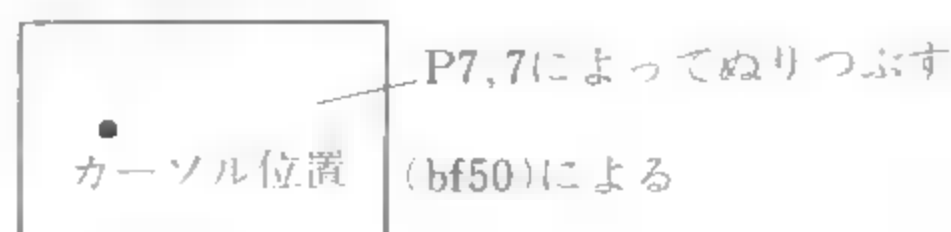
【解 説】 ◎ 塗りつぶしと回転

- ① 3 行目 4 行目で四角形をかきます。
- ② 5 行目で右下へカーソルを移します。
- ③ 次の C7, P7, 7 は次の意味です。

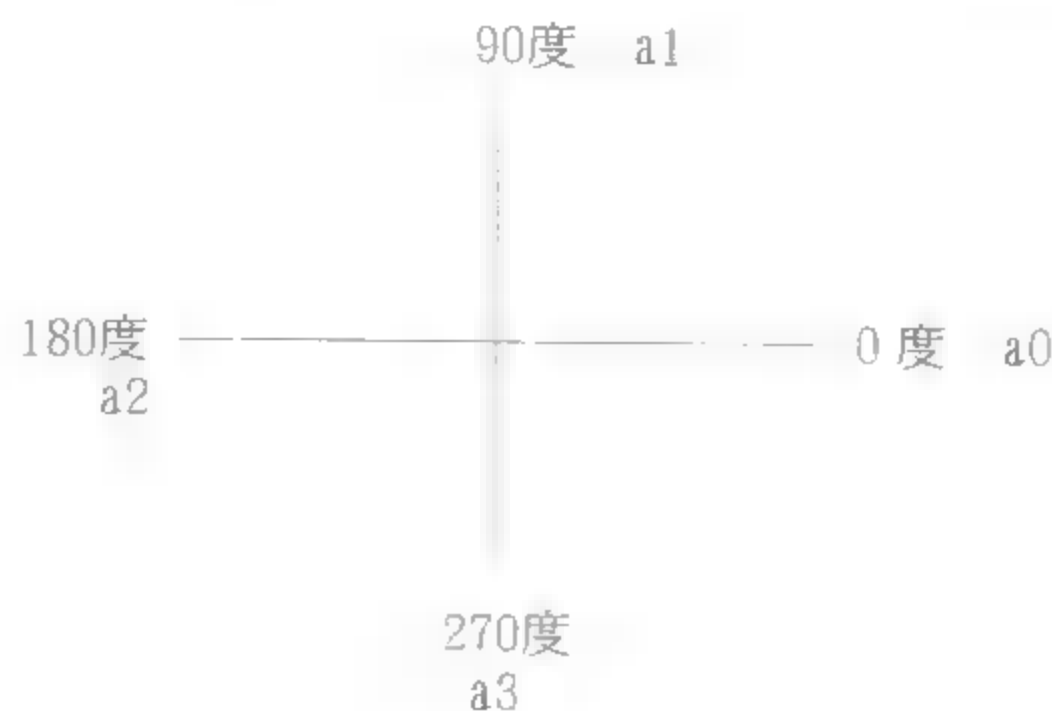
C は色指定です。COLOR, PALETTE および SCREEN 命令と関連します。ここでは 7 によって白を示します。

P は図形の塗りつぶしを意味します。最初の 7 は塗りつぶす色で白、次の 7 は塗りつぶす境界色で白です。すなわち、現在カーソルのある領域で、境界が白線の中を白でぬりつぶします。

3～4行目の図



- ④ 7 行目～10 行目の a は回転角を示し数値は角です。角は 0, 90, 180, 270 度でのおの 0, 1, 2, 3 で表わします。次図のようになります。

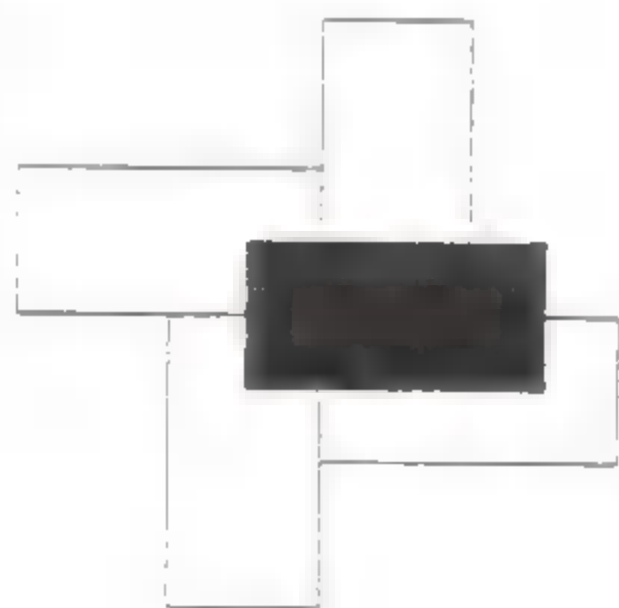


- ⑤ 7～10 行目は四角形を回転してかきます。

【プログラム例】

```
SCREEN 0
CLS 0
DRAW "s2"——— 四角形をかく
DRAW "d100 r200 u100 1200"———
DRAW "bf50"——— カーソルを四角形の中へ入れる
DRAW "c7 p7,7"——— 塗りつぶし
DRAW "a0 d100 r200 u100 1200"——— 回転した四角形を4つかく
DRAW "a1 d100 r200 u100 1200"———
DRAW "a2 d100 r200 u100 1200"———
DRAW "a3 d100 r200 u100 1200"———
END
```

【結 果】



【例題 26】 DRAW を使った図

四角形をかき次に 60 度反時計方向と 180 度時計方向へ回転した図でかくプログラムをつくれ。

【解 説】 ◎ 任意の角の回転をします。

- ① 3～4 行目で四角形をかきます。
- ② 5 行目と 6 行目の `ta` は回転を示します。数値は-360 から 360 までの整数で、正のとき反時計まわり、負のとき時計まわりとなります。

したがって 5 行目の `ta60` は反時計まわりに 60 度、6 行目の `ta-180` は時計まわりに 180 度の回転を示します。

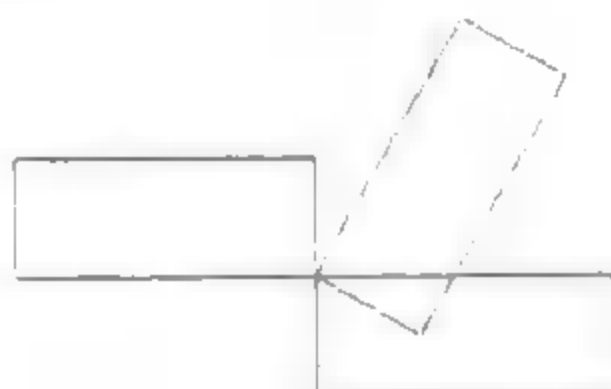
【プログラム例】

```

SCREEN 0
CLS 0
DRAW "s2"————四角形の表示
DRAW "d100 r200 u100 1200"————
DRAW "ta60 d100 r200 u100 1200"————60 度反時計方向へ回転した図
DRAW "ta-180 d100 r200 u100 1200"————180 度時計方向へ回転した図
END

```

【結 果】



【例題 27】 DRAW を使った図

DRAW を使って、四角形をかき、次に長さが 2 倍、3 倍の図をかくプログラムをつくれ。

【解 説】 ◎ 拡大図をつくります。

- ① 3 行目で縦 100、横 200 の四角形をつくります。
- ② 4 行目で S2 の指定により、縦、横とも長さが 2 倍の図をつくります。
- ③ 5 行目で S3 の指定により、縦、横とも長さが 3 倍の図をつくります。

【プログラム例】

```

SCREEN 0
CLS 0
DRAW "s1 d100 r200 u100 1200"————四角形をかく
DRAW "s2 d100 r200 u100 1200"————2 倍の四角形をかく
DRAW "s3 d100 r200 u100 1200"————3 倍の四角形をかく
END

```

【結 果】



【例題 28】 DRAW を使った図

DRAW を使って【結 果】の図をかけ。

【解 説】 ◎ 部分文字列コマンドを使います。

- ① 3 行目と 6 行目の組み合わせは次のようになります。

3 行目 a1\$= "S2"

→DRAW "S2"

6 行目 DRAW "x"+VARPTR\$(a1\$)

- ② 同様に 5 行目と 7 行目の組合わせは次のとおりになります。

5 行目 a3\$="d100r100u100l100" →DRAW"d100 r100 u100 l100"

7 行目 DRAW"⌵"+VARPTR\$(a3\$)

- ③ 10 行目と 11 行目の組合わせは次のようになります。

10 行目 Z=1 →DRAW "S1"

11 行目 DRAW "S="+VARPTR\$(Z)

【プログラム例】

```
SCREEN 0
CLS 0
a1$ = "s2"
a2$ = "s3"
a3$ = "d100 r100 u100 l100"
DRAW "x" + VARPTR$(a1$)
DRAW "x" + VARPTR$(a3$)
DRAW "x" + VARPTR$(a2$)
DRAW "x" + VARPTR$(a3$)
z = 1
DRAW "s=" + VARPTR$(z)
DRAW "x" + VARPTR$(a3$)
END
```

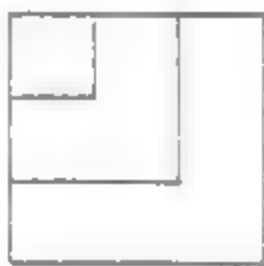
変数に文字列を代入

DRAW 文

変数に数値を代入

DRAW 文

【結 果】



〔演習問題〕

- (23) 画面左上に"日"をかき、画面のドットをチェックして、表示のとき 100 ドット桁プラス、200 ドット桁プラスの位置のドットを表示することで"日"を右に 2 つコピーするプログラムをつくれ。
- (24) "S2"を指定して縦 100、横 300 の長方形をかき、次に a0, a1, a2, a3 を指定して回転した図をつくるプログラムをつくれ。
- (25) 結果の図をかくプログラムをつくれ。
- (26) 結果の図をかくプログラムをつくれ。
- (27) 結果の図をかくプログラムをつくれ。

【解 答】

(23) 【プログラム例】

```

SCREEN 0
CLS 0
LINE (10, 10)-(50, 50), , B ———— "日"の表示
LINE (10, 30)-(50, 30) ————
FOR i = 0 TO 60 ————
  FOR j = 0 TO 60 ————
    IF POINT(i, j) = 7 THEN ———— 各ドットをチェックして
      PSET (i + 100, j) ———— 表示してあるとき
      PSET (i + 200, j) ———— 100 ドット桁プラス、200 ドット桁
                                プラスの位置のドットを表示
    END IF
  NEXT j
NEXT i
END

```

【結 果】



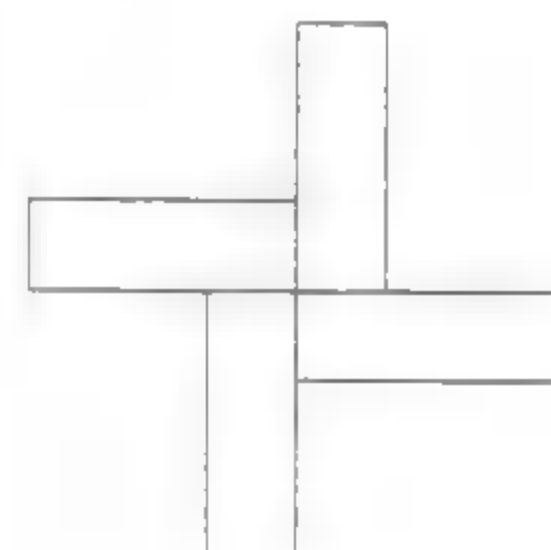
(24) 【プログラム例】

```

SCREEN 0
CLS 0
DRAW "s2" ———— 長方形の表示
DRAW "d100 r300 u100 l300" ————
DRAW "a0 d100 r300 u100 l300" ———— 回転した図の表示
DRAW "a1 d100 r300 u100 l300" ————
DRAW "a2 d100 r300 u100 l300" ————
DRAW "a3 d100 r300 u100 l300" ————
END

```

【結 果】



(25) 【プログラム例】

```

SCREEN 0
CLS 0
DRAW "bm300, 100" ———— カーソルを(300,100)
DRAW "s2" ———— へ移動
DRAW "f100 r100 g100 d100" ———— 図の表示
DRAW "l100 h100 u100 e100" ————
END

```

【結 果】



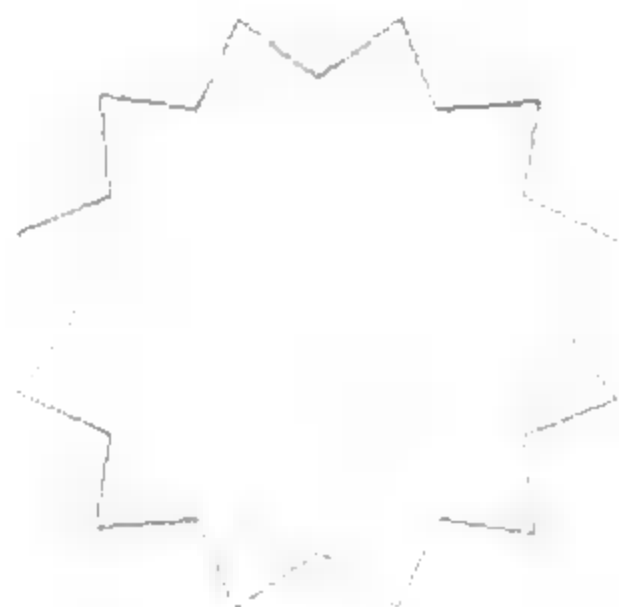
(26) 【プログラム例】

```

SCREEN 0
CLS 0
DRAW "s1"
DRAW "bm300,300"———————カーソルを(300,300)へ移動
FOR i = 0 TO 360 STEP 30 ————30度ずつ回転した図を表示
    DRAW "ta=" + VARPTR$(i)
    DRAW "r100 u100"
NEXT i
END

```

【結果】



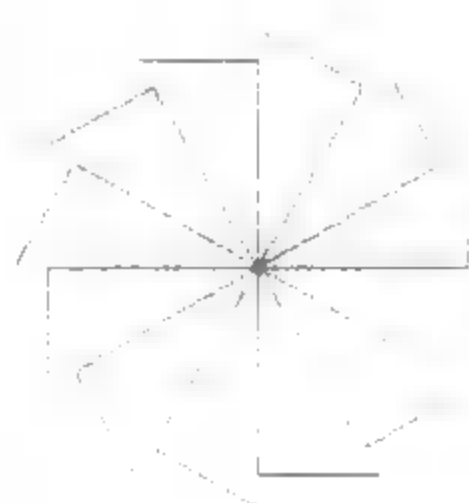
(27) 【プログラム例】

```

SCREEN 0
CLS 0
DRAW "s1"
DRAW "bm300,200"———————カーソルを(300,200)へ移動
FOR i = 0 TO 360 STEP 30 ————30度ずつ回転した図を表示
    DRAW "ta=" + VARPTR$(i)
    DRAW "r200 u100"
    DRAW "bm300,200"
NEXT i
END

```

【結果】



IV 応用編



1 章 N₈₈BASIC (86) との互換

【応用問題1】 表 示

N₈₈BASIC(86)でつくった次のプログラムをQuick BASICで実行せよ.

```
10 PRINT "クイックヘーシック"  
20 END
```

【解 説】 ① N₈₈BASIC(86)のプログラムをQuick BASICで実行します.

- ① N₈₈BASIC(86)を使って、問題のプログラムをつくります. これをアスキーファイルとして格納します.

save "PRAC-1.BAS", a によってアスキーセーブします.

- ② Quick BASICを走らせます. DIRによってPRAC-1.BASがあることを確認します.

RRR-82	BAS	RRR-85	BAS	HAIRETSU	BAS	HAIRETSU	IN	RRR-86	BAS
PAI	IN	RRR-87	BAS	RRR-88	BAS	KAMOKU	IN	RRR-89	BAS
RRR-90	BAS	RRR-95	BAS	RRR-96	BAS	RRR-97	BAS	RRR-98	BAS
RENSHU1	BAS	PRAC-1	BAS						

142 個のファイルがあります.
651264 バイトが使用可能です.

- ③ ファイルメニューを選択し, , PRAC-1. によってファイルPRAC-1.BASを読み込みます.

PRAC-1.BAS

```
10 PRINT "クイックヘーシック"  
20 END
```

- ④ このプログラムはこのまま実行できます. によってプログラムを実行します.

クイックヘーシック

【プログラム例】

```
10 PRINT "クイックヘーシック"  
20 END
```

【応用問題2】 桁・行の指示

次のプログラムを N₈₈BASIC(86)でつくり、これを Quick BASIC で実行せよ。

```
10 CLS
20 FOR I = 1 TO 10
30     LOCATE I, 10: PRINT "A"
40 NEXT I
50 END
```

【解 説】 ◎ N₈₈BASIC(86)でつくったプログラムを Quick BASIC で一部変更して実行します。

① N₈₈BASIC(86)を使って問題のプログラムをつくります。

② これをアスキーセーブします。

save "PRAC-2.BAS", a です。

③ Quick BASIC を走らせファイルメニューから **O**, PRAC-2 **J** によりこのプログラムを読み出します。

```
10 CLS
20 FOR I = 1 TO 10
30     LOCATE I, 10: PRINT "A"
40 NEXT I
50 END
```

④ **f-5** で実行すると次のように縦に "A" を表示します。

```
A
A
A
A
A
A
A
A
A
A
A
```

⑤ 元のプログラムでは横に "A" を表示するものですから、縦横が逆です。これは LOCATE x, y が N₈₈BASIC(86)では x が桁, y が行を示すのに対し, Quick BASIC では x が行, y が桁を示すからです。したがって、桁と行を逆に修正します。

【プログラム例】

```
10 CLS
20 FOR I = 1 TO 10
30     LOCATE 10, I: PRINT "A"          LOCATE i,10 を LOCATE 10,i に変更
40 NEXT I
50 END
```

【結 果】

```
AAAAAAAAAA
```

【応用問題3】 グラフィック

次のプログラムを N₈₈BASIC(86)でつくり、次にこれを Quick BASIC で実行せよ。

```

10 SCREEN 2, 0, 0, 1
20 CLS 3
30 LINE (200, 50)-(400, 250), , B
40 LINE (200, 50)-(400, 250)
50 LINE (200, 250)-(400, 50)
60 LINE (100, 250)-(500, 350), , BF
70 END

```

【解 説】 ① N₈₈BASIC(86)でつくったグラフィックスを Quick BASIC で実行します。

① 問題のプログラムを N₈₈BASIC(86)でつくります。

② これをアスキーセーブします。

save "PRAC-3.BAS", a です。

③ Quick BASIC でこれを読み出します。ファイルメニューから **[O]**、PRAC-3.BAS **[J]** です。

④ これをそのまま **[f-5]** とするとエラーとなります。

SCREEN は、N₈₈BASIC(86)と Quick BASIC では、使い方が違います。Quick BASIC の SCREEN 文の 2 番目は 8086/v30, 80286 を CPU とするマシンでは指定できません。

⑤ 行番号 10 と 20 をつぎのように修正します。

PRAC-4.BAS

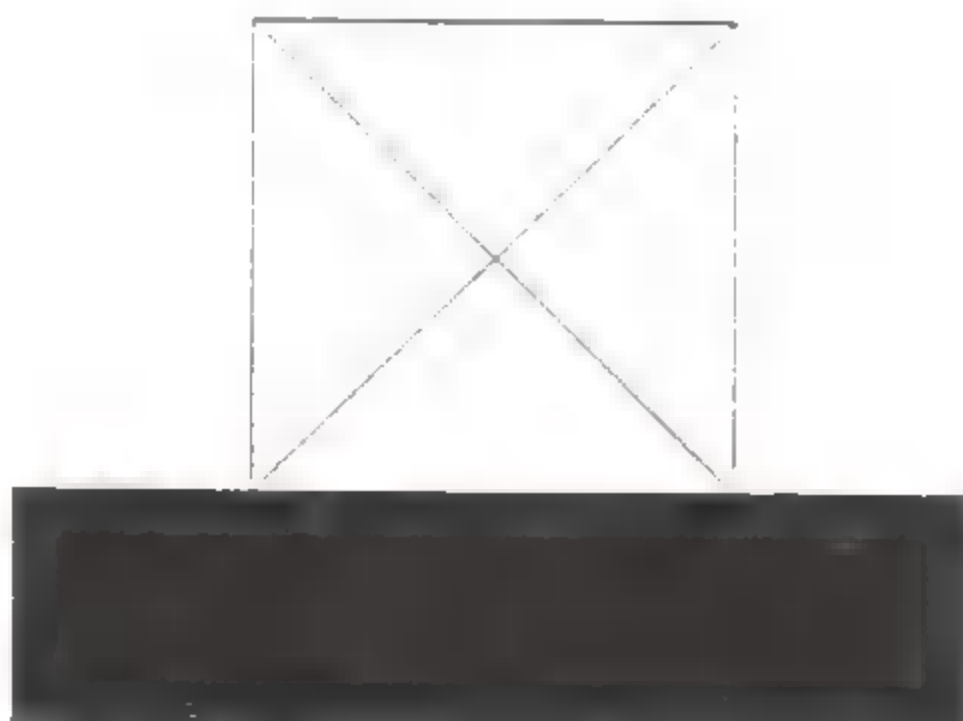
```

10 SCREEN 0
20 CLS
30 LINE (200, 50)-(400, 250), , B
40 LINE (200, 50)-(400, 250)
50 LINE (200, 250)-(400, 50)
60 LINE (100, 250)-(500, 350), , BF
70 END

```

⑥ **[f-5]** で実行します。

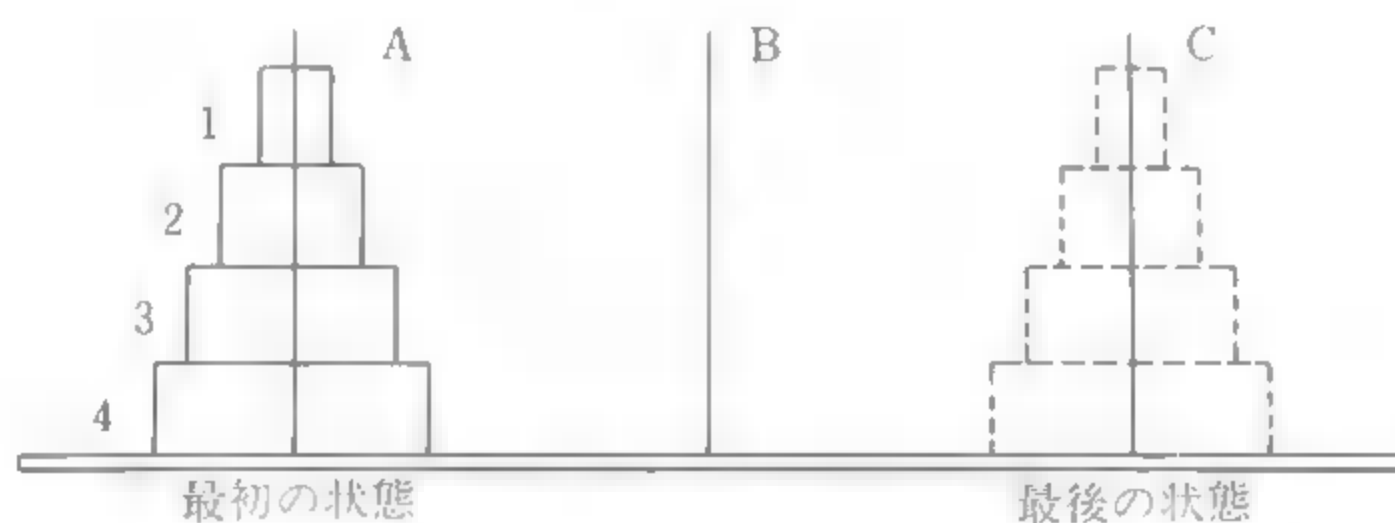
【結 果】



2章 再帰プログラム

【応用問題4】 ハノイの塔

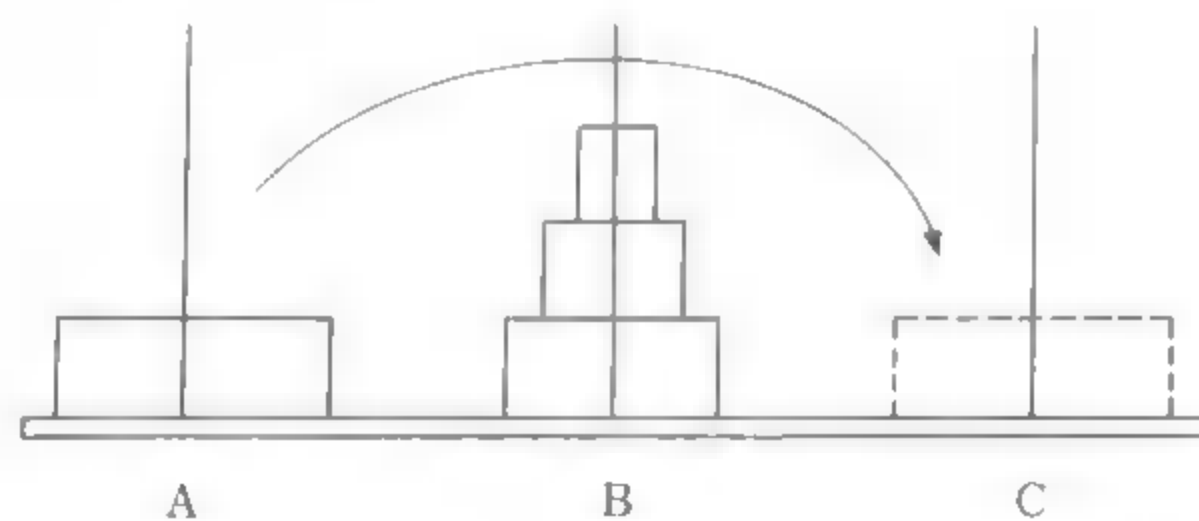
再帰プログラムを利用してハノイの塔を解くプログラムをつくれ。ここでの例は次のような問題とします。



1～4の円板（円筒）、A～Cの柱があります。Aの柱に大きい円板を下にして1～4の円板をさしこみます。これを1回に1つの円板を動かし、小さい円板の上に大きい円板が乗らないことを条件にしてCの柱に大きい順に並べなおしてください。

【解 説】 ① ハノイの塔の問題を再帰プログラムで解きます。

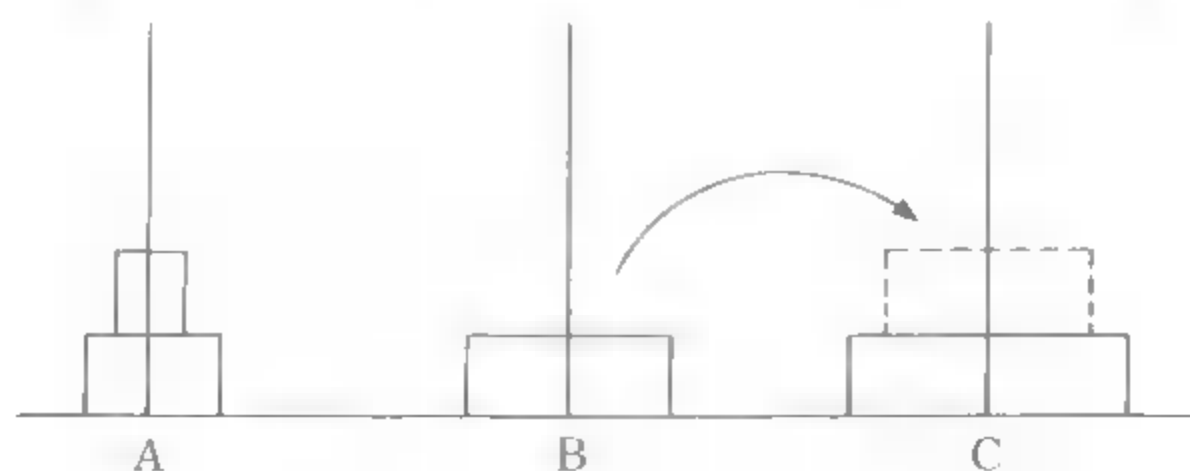
- ① ハノイの塔の問題は複雑そうですが、再帰という考えを入れると簡単に解けます。この考え方をまず説明します。説明をみる前に自分でも実際にやってみてください。
- ② 一番大きい円板がAの柱からCの柱へ動くときの状態を考えてください。次のようになるはずです。



Aの上には大きい円板以外何もない
Cの上には何もない
したがって、Bに残りが大きい順に並んでいる

“Bに残りが大きい順に並んでいる”ことがポイントです。

- ③ 次に残りの円板のうち最大のものが、Cの円板の上へ乗る直前の状態は次のとおりです。

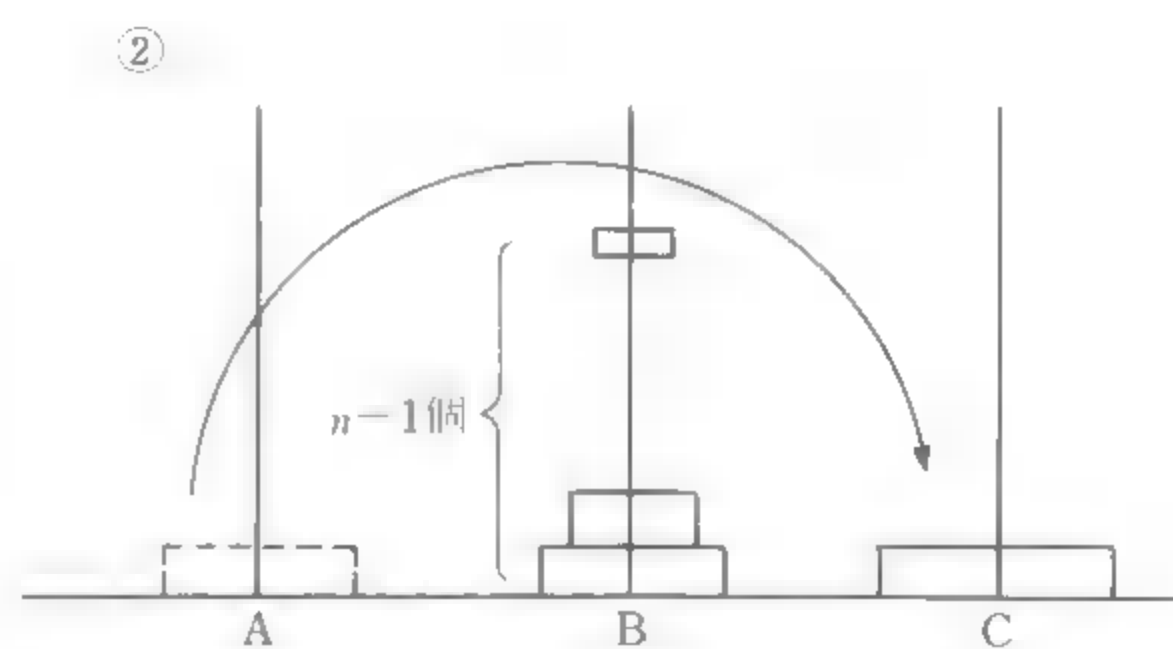
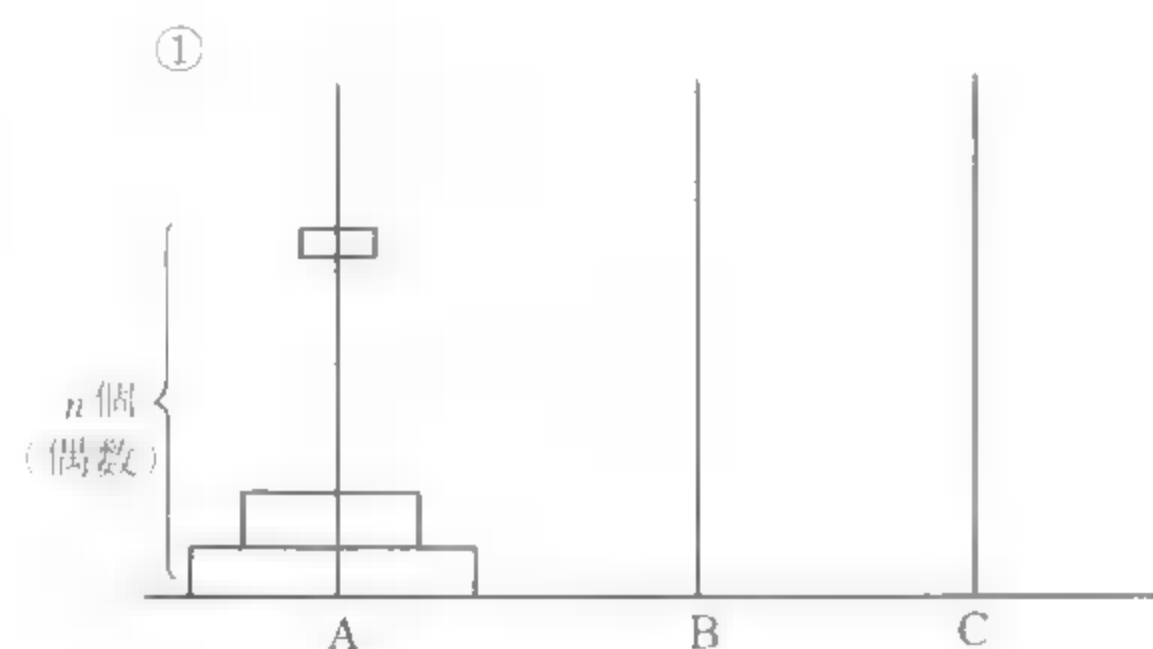


Aの柱には残りが大きい順に並んでいる
Bの上には3つのうちの最大のものが残る
Cは4つのうちの最大のものの以外はない

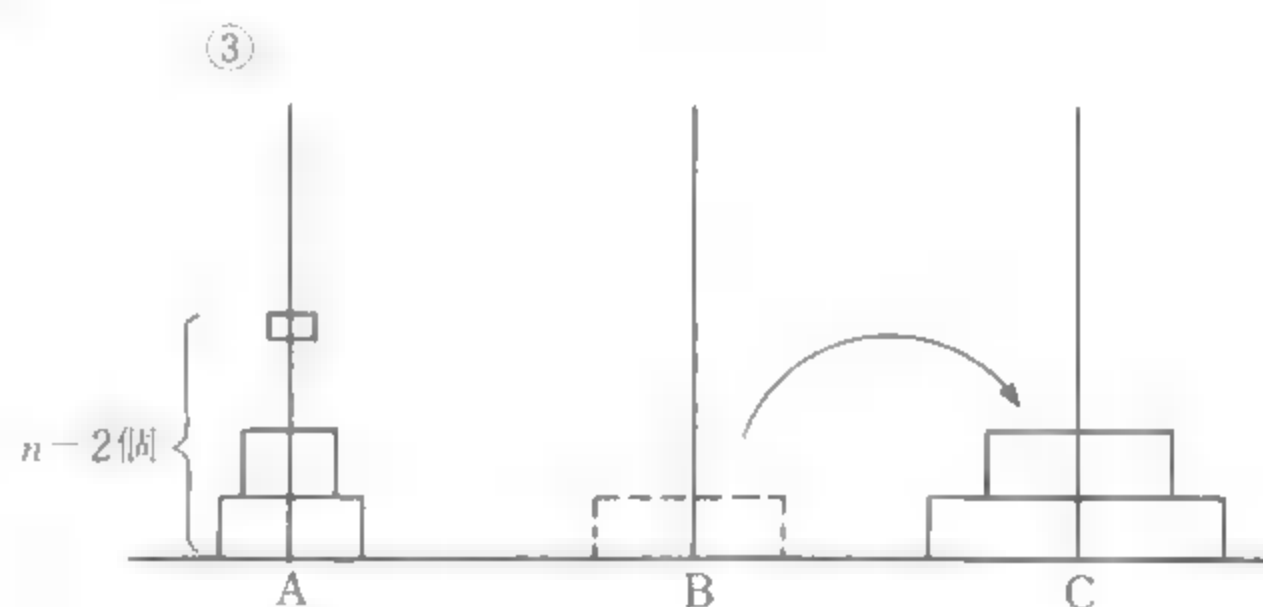
Aに残りが大きい順に並んでいることがポイントです。

- ④ すでに気づいたとおり、 n 個の中の最大のものを動かすときは動かす先は n より大きいものが順

に並んでいるか何もない状態であり、他の柱は $n-1$ 個の円板が大きい順に並んでいます。ということは n 個のものを移すことは $n-1$ 個のものを移すことと同じことであり、 $n-1$ 個のものを移すことは $n-2$ 個のものを移すのと同じことです。



n 個のものを移すことは最大の円板を移した後は $n-1$ 個のものをとなりの柱から移すのと同じである。このくり返しとなります。



⑤ 再帰的に考えると次のようになります。

```
SUB hanoi(x%,a$,b$,c$)      ...x%を4, a$を"A", b$を"C", c$を"B"としてhanoiを呼
                              ...び出します。
  1F x%>0 THEN
    hanoi x%-1,a$,c$,b$      ...n-1個の円板をaからbへ
    PRINT x%;"ノエンバンオ";a$"カラ";b$;"へ"  ...最大のものをaからcへ
    hanoi x%-1,c$,b$,a$
  END IF
END SUB
```

【プログラム例】

```
DECLARE SUB hanoi (x%, a$, b$, c$)
hanoi 4, "A", "C", "B"
END

SUB hanoi (x%, a$, b$, c$)
  IF x% > 0 THEN
    hanoi x% - 1, a$, c$, b$
    PRINT x%; "ノエンバンオ"; a$; "カラ"; b$; "へ"
    hanoi x% - 1, c$, b$, a$
  END IF
END SUB
```

【結 果】

```

1 ノ エンハ`ン オ A カラ B ヘ
2 ノ エンハ`ン オ A カラ C ヘ
1 ノ エンハ`ン オ B カラ C ヘ
3 ノ エンハ`ン オ A カラ B ヘ
1 ノ エンハ`ン オ C カラ A ヘ
2 ノ エンハ`ン オ C カラ B ヘ
1 ノ エンハ`ン オ A カラ B ヘ
4 ノ エンハ`ン オ A カラ C ヘ
1 ノ エンハ`ン オ B カラ C ヘ
2 ノ エンハ`ン オ B カラ A ヘ
1 ノ エンハ`ン オ C カラ A ヘ
3 ノ エンハ`ン オ B カラ C ヘ
1 ノ エンハ`ン オ A カラ B ヘ
2 ノ エンハ`ン オ A カラ C ヘ
1 ノ エンハ`ン オ B カラ C ヘ

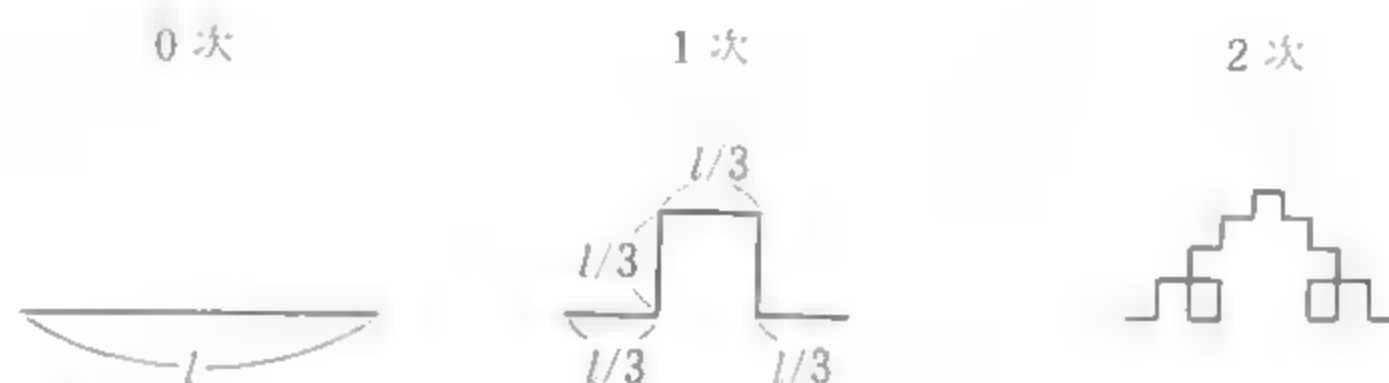
```

【応用問題5】 ステッチ

再帰を使って、結果のステッチ模様をかくプログラムをつくれ。

【解 説】

- ① クロスステッチはコッホ曲線を使ってかきます。コッホ曲線は次のようにつくります。



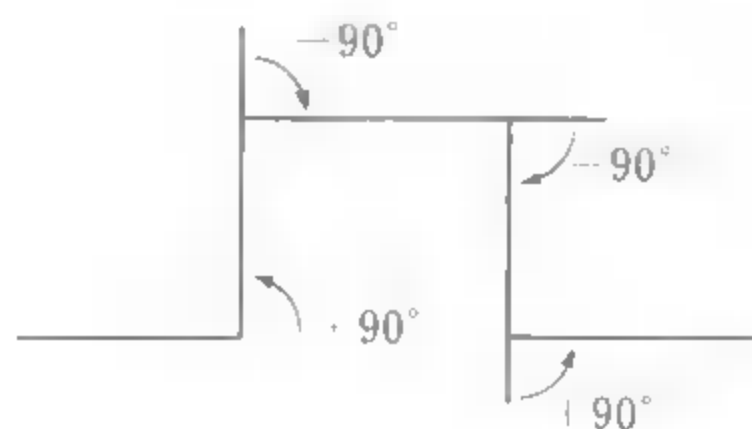
0 次は直線長さ l

1 次は左から $1/3$ のところで $1/3$ の正方形（ただし、下辺はなし）をつくる

2 次は 1 次の各直線部で $1/3$ のところすなわち $1/9$ のところで正方形をつくる

- ② 1 次のコッホ曲線は 0 次のコッホ曲線を 5 本かいたものです。2 次のコッホ曲線は 1 次のコッホ曲線を 5 本かいたものです。一般化すると n 次のコッホ曲線は $n-1$ 次のコッホ曲線を 5 本かけばよいことになります。

このとき正方形の方向は次のように $+90^\circ$, -90° , -90° , $+90^\circ$ 度です。



したがって、プログラム例のサブルーチン `stech` がコッホ曲線をかき再帰呼び出しを行うルーチンです。

【プログラム例】

```

DECLARE SUB stech (n!, l)
DECLARE SUB move (l)
DECLARE SUB kaku (deg!)
COMMON SHARED xx, yy, a

```



```

SCREEN 0
CLS 0
xx = 300
yy = 350
a = 45
n = 4
l = 3
FOR i = 1 TO 4
    stech n, l
    kaku 90
NEXT i
END

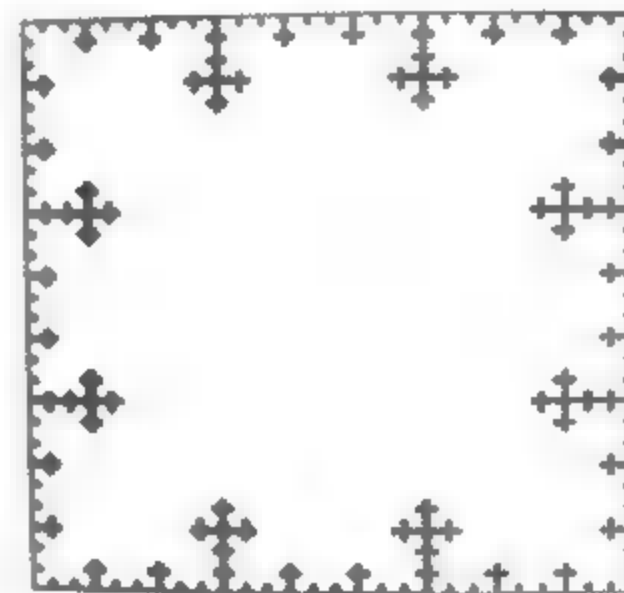
SUB kaku (deg)
    a = a + deg
    a = a MOD 360
END SUB

SUB move (l)
    x = l * COS(3.14159 / 180 * a)
    y = l * SIN(3.14159 / 180 * a)
    LINE (xx, yy)-(xx + x, yy + y)
    xx = xx + x
    yy = yy + y
END SUB

SUB stech (n, l)
    IF n = 0 THEN
        move l
    ELSE
        stech n - 1, l
        kaku -90
        stech n - 1, l
        kaku 90
        stech n - 1, l
        kaku 90
        stech n - 1, l
        kaku -90
        stech n - 1, l
    END IF
END SUB

```

【結 果】



3章 円弧を使ったグラフ

【応用問題6】 円帯グラフ

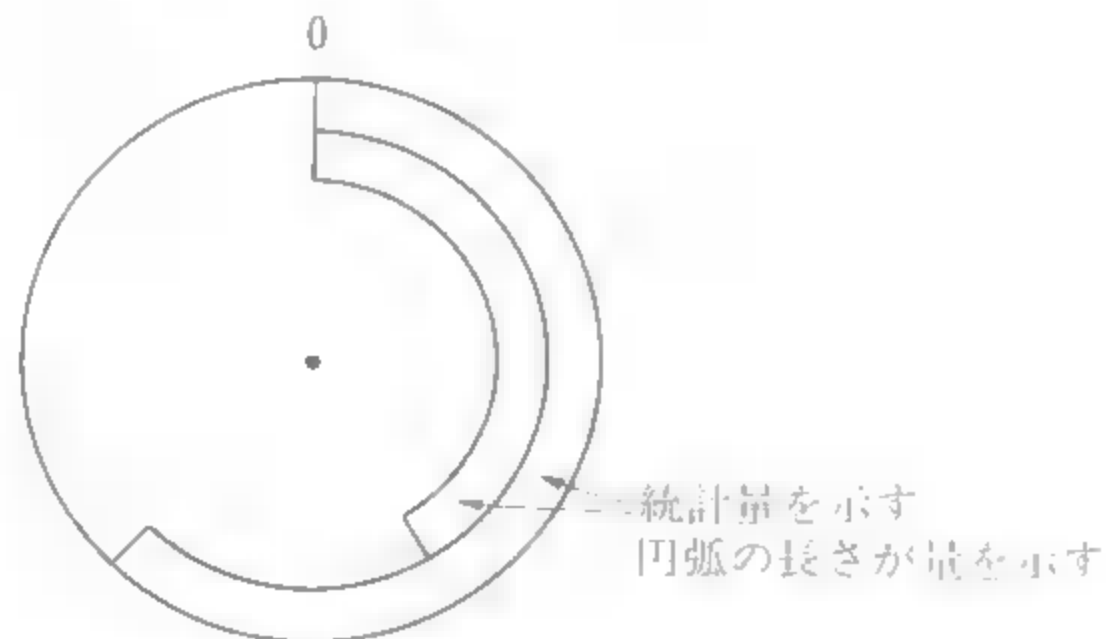
下の表のデータを円帯グラフとするプログラムをつくれ。

1987年商品別売上高

品 名	売上高（億円）
家 電	250
モーター	220
コントローラー	150
エレクトロニクス	360

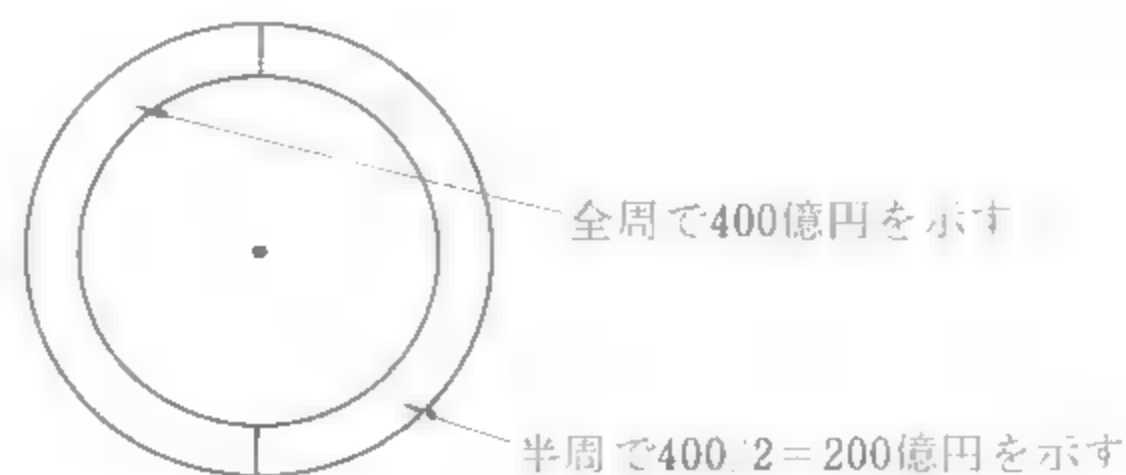
【解 説】 ① 円帯グラフをつくります。

- ① 下図のように円周上の円弧の長さで統計量を表わすグラフを円帯グラフと名づけます。

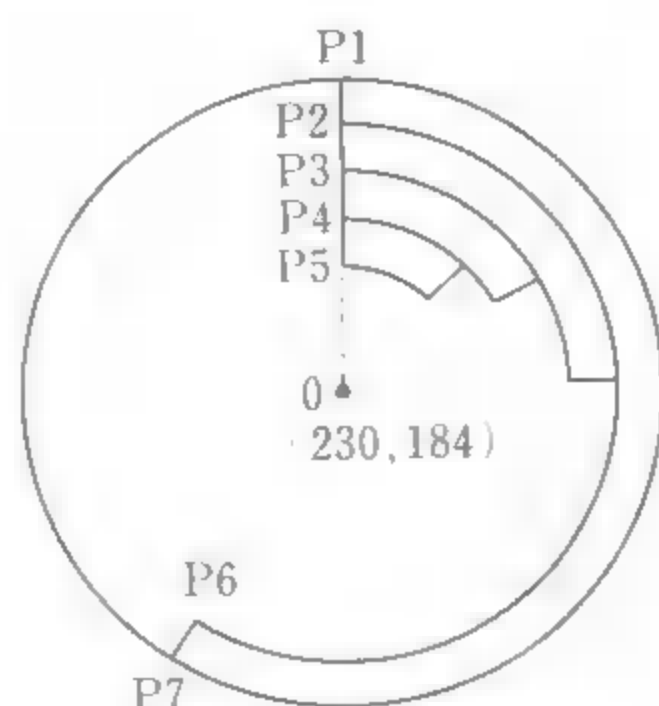


円弧の長さは 360 度で比率なら 100%, 量ならグラフで表示できる最大量とします。

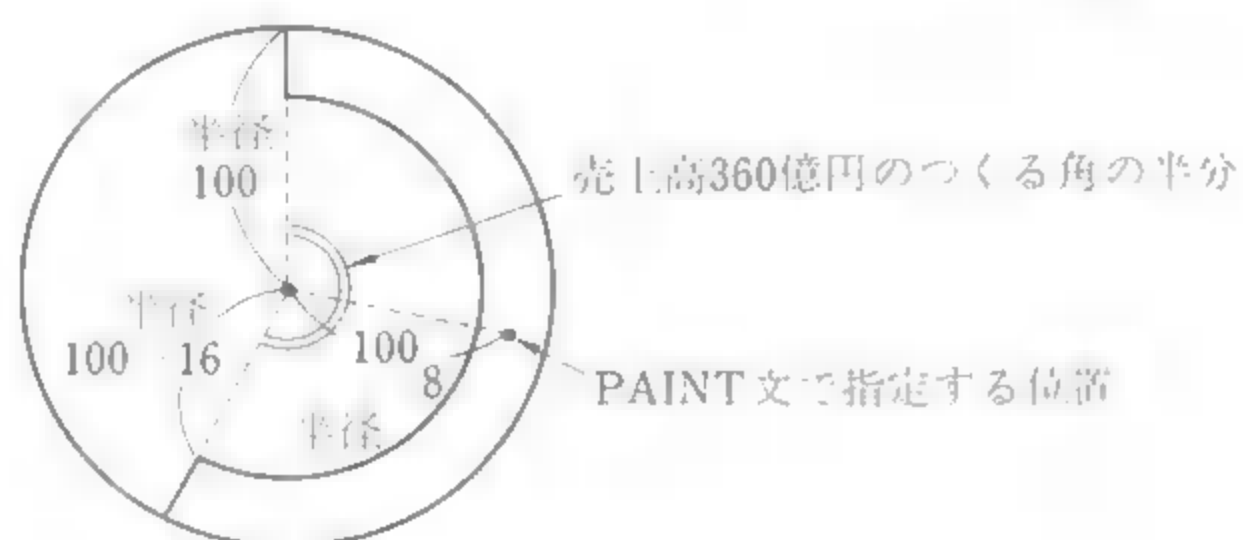
- ② 4 つの統計量を大きさの順に並べなおし、量の多いものを外周側へおきます。
 ③ 売上高は 360 億円が最大ですから、ここでは全周すなわち 360 度の場合 400 億円を示すものとします。



- ④ したがって、X 億円の売上は $X/400 \times 2 \times 3.14159$ ラジアンで表示できます。400 億円の場合 2×3.14159 ラジアンすなわち 360 度です。
 ⑤ 図の表示は次のようになります。



- ① 中心を(230,184), 半径を100とする円をかく
- ② 半径上のP1とP5を結ぶ。P1は半径100の円周上の点で時計の12時の位置です。P5はP1と中心を結ぶ半径上の点でP1とP5の長さは4本の帯の幅です。今回は1本の帯の幅は16ドットとしています。したがって、P1-P5はY方向16*4=64ドットです。
- ③ P2-P6の円弧をかきます。中心を(230,184)とし、半径は(100-16)ドットで、P2-0-P6のつくる角度はエレクトロニクスの売上高360/400*2*3.14159ラジアンです。
- ④ P6とP7を結びます。P7は中心を(230,184)とし、半径を100とする円周上で、P1-0-P7のつくる角がエレクトロニクスの売上高360/400*2*3.14159ラジアンです。
- ⑤ 帯の中の1点を指定してPAINT文でハッチングします。帯の中の1点は中心を(230,184)とし、半径を(100-8)ドット、角度をエレクトロニクスの売上高360/400/2*2*3.14159ラジアンの位置です。



【プログラム例】

```

DECLARE SUB aa ()
DECLARE SUB bb ()
COMMON SHARED a$(), a(), b$()
DIM a$(1 TO 4), a(1 TO 4), b$(1 TO 4)
SCREEN 0
CLS 0
DEF fnx (c, r, x) = c + r * SIN(x) ———— 円周上の点の X 座標値の定義
DEF fny (c, r, y) = c - r * COS(y) ———— 円周上の点の Y 座標値の定義
DEF fnd (d) = d / 400 * 2 * 3.14159 ———— 売上高に相当する中心角を求める式の定義
b$(1) = CHR$(&HFF) ———— ハッチングパターンの登録
b$(2) = CHR$(&H11)
b$(3) = CHR$(&H22) + CHR$(&H11)
b$(4) = CHR$(&HC) + CHR$(&HAA)
FOR i = 1 TO 4 ———— 商品名と売上の読み込み
    READ a$(i), a(i)
NEXT i
FOR i = 1 TO 3 ———— 売上高の大きい順の並べ替え
    FOR j = i + 1 TO 4
        IF a(i) > a(j) THEN GOTO a:
        SWAP a$(i), a$(j)
        SWAP a(i), a(j)
a: NEXT j
NEXT i

```

```

CALL aa ----- グラフ準備ルーチンへ分岐
CALL bb ----- グラフ作成ルーチンへ分岐
END
DATA カデシ, 250, モータ, 220, コントローラ, 150
DATA エレクトロニクス, 360

```

```

SUB aa ----- グラフ準備ルーチン
  LOCATE 1, 20: PRINT "ウリアゲタカ(1987年)" ----- 標題
  FOR i = 128 TO 224 STEP 32 ----- 凡例のパターン
    LINE (400, i)-(440, i + 16), , B
    PAINT (420, i + 8), b$((i - 96) / 32)
  NEXT i
  FOR i = 8 TO 14 STEP 2 ----- 凡例の商品名
    LOCATE i, 60: PRINT a$(i / 2 - 3)
  NEXT i
  LOCATE 4, 29: PRINT "0 オフイン" ----- 単位, 売上高の値
  LOCATE 3, 26: PRINT 400
  LOCATE 11, 42: PRINT 100
  LOCATE 19, 26: PRINT 200
  LOCATE 11, 10: PRINT 300
END SUB

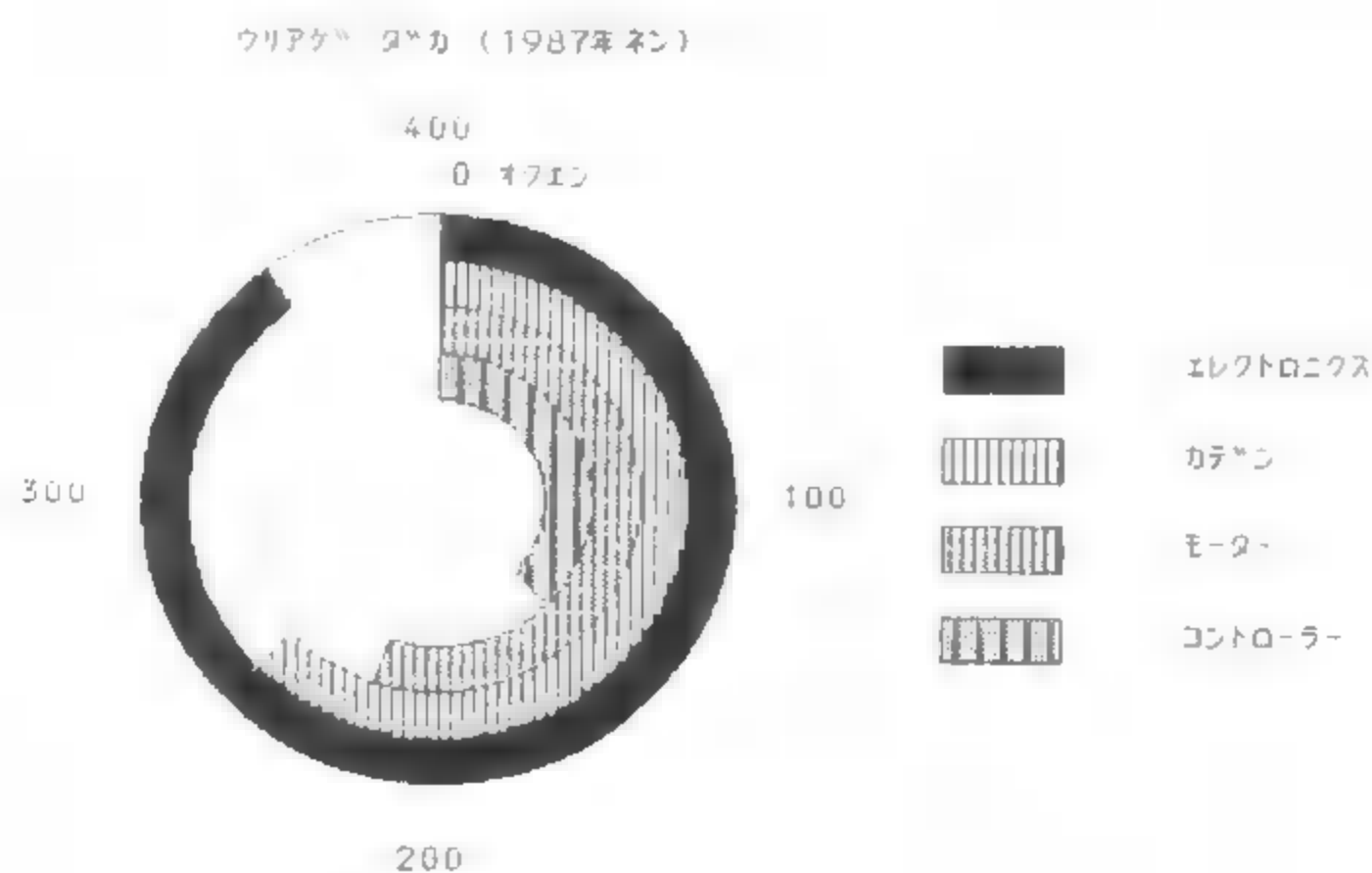
```

```

SUB bb ----- グラフの作成ルーチン
  CIRCLE (230, 184), 100 ----- 円
  LINE (230, 84)-(230, fny(184, 100 - 4 * 16, fnd(0))) ----- 基準線
  FOR i = 1 TO 4 ----- 4 データの帯
    PSET (fnx(230, 100 - i * 16, fnd(0)), fny(184, 100 - i * 16, fnd(0)))
    FOR q = 0 TO fnd(a(i)) + .03 STEP .05 ----- 売上高に相当する円弧
      x1 = fnx(230, 100 - i * 16, q)
      y1 = fny(184, 100 - i * 16, q)
      LINE -(x1, y1)
    NEXT q
    x2 = fnx(230, 100 - (i - 1) * 16, fnd(a(i))) ----- 帯の最終端の直線
    y2 = fny(184, 100 - (i - 1) * 16, fnd(a(i)))
    LINE -(x2, y2)
    x3 = fnx(230, 100 - (i - 1) * 16 - 8, fnd(a(i) / 2)) ----- ハッチング
    y3 = fny(184, 100 - (i - 1) * 16 - 8, fnd(a(i) / 2))
    PAINT (x3, y3), b$(i)
  NEXT i
END SUB

```

【結果】



【応用問題 7】 両側円帯グラフ

次表を両側円帯グラフとするプログラムをつくれ.

男女別成績

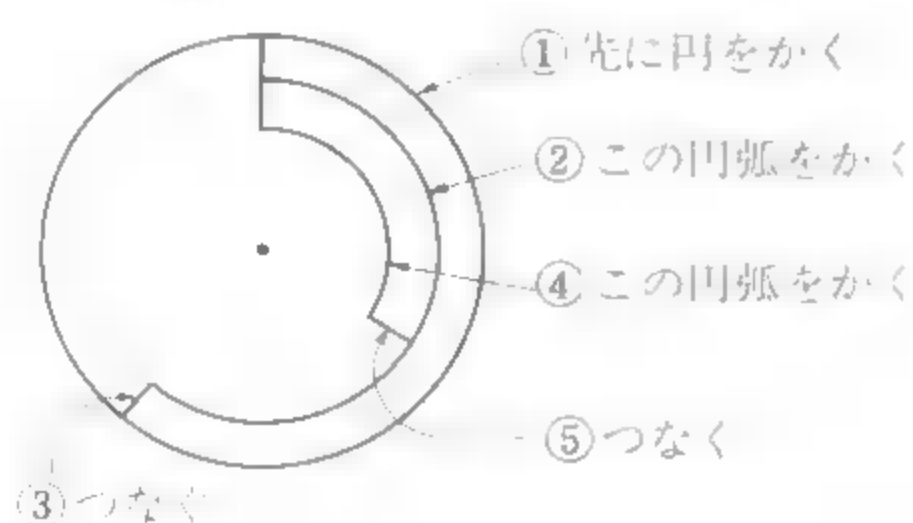
課目	男	女
英語	85	70
国語	75	85
社会	80	70
数学	88	75

【解 説】 ◎ 両側円帯グラフをつくります.

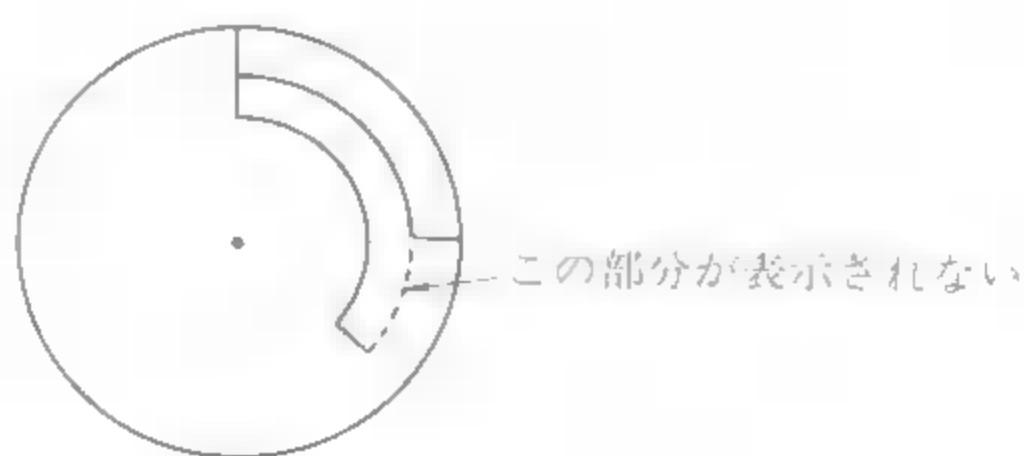
- ① 【応用問題 6】でつくった帯グラフを応用し、基準線から両側に π ラジアンを 100 点として帯を表示します. これを両側円帯グラフと名づけます.
- ② ここではデータの大きさ順の並べかえはしません. 基準線から右側, 時計まわりを男子の成績, 左側, 反時計まわりを女子の成績とします.
- ③ 右側の考え方は【応用問題 6】とほぼ同じですから, その解説をみてくざたい. ただし, データの並べかえがしてありませんので, データが外側より内側の方が大きい場合その帯の外側がかけないこととなりますので, 帯の両側をかくことにします.

【応用問題 6】のかき方

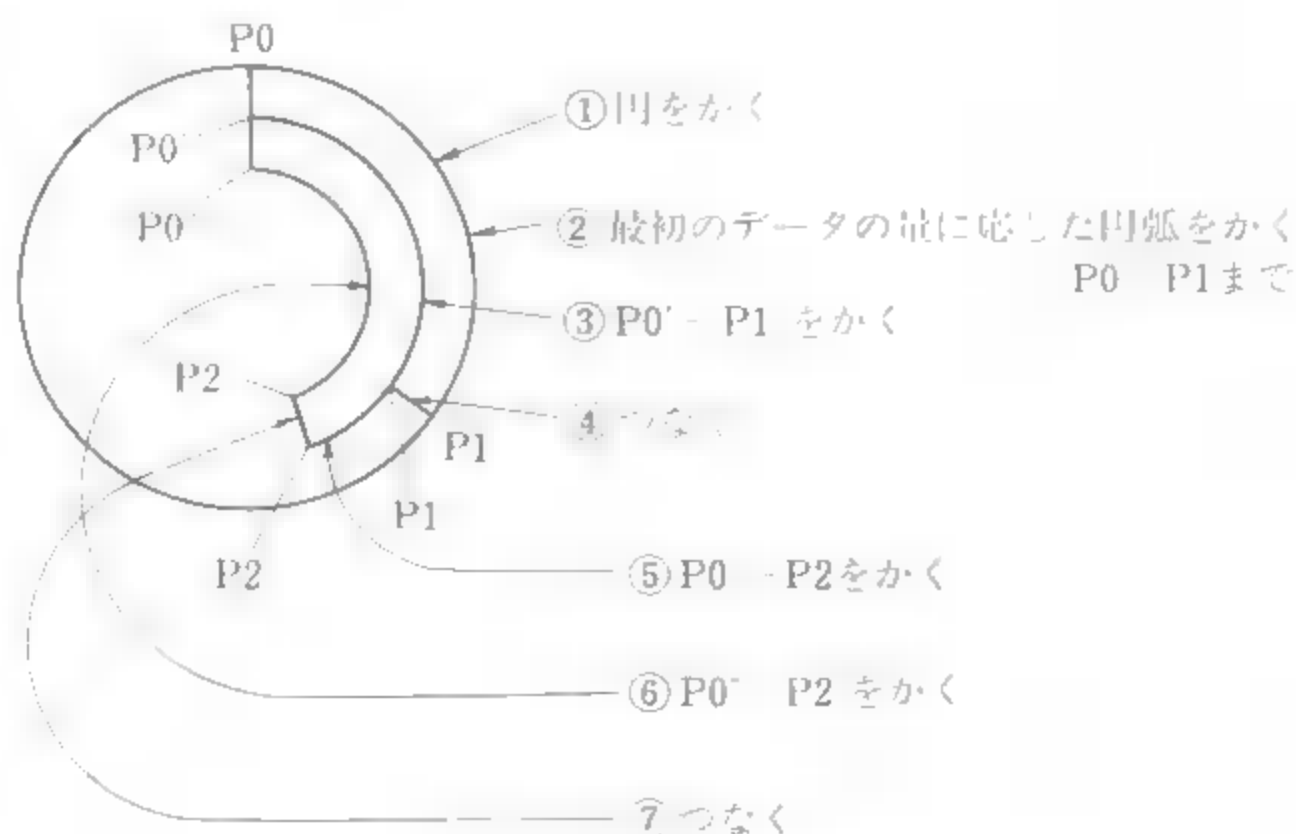
データが外側の方が大きい場合



データが内側の方が大きい場合



今回のかき方



- ④ 左側の考え方は右側と同じですが, 座標値の表わし方が逆となります.
右側からかく場合の X 座標値は $C+R\sin(X)$, Y 座標値は $C-R\cos(X)$ です.

左側からかく場合は X 座標値は $C + R \cos(X + 3.14159/2)$, Y 座標値は $C - R \sin(X + 3.14159/2)$ です。

【プログラム例】

```

DECLARE SUB aa ()
DECLARE SUB bb ()
DECLARE SUB cc ()
DECLARE SUB dd ()
COMMON SHARED a$(), a(), b$()
DIM a$(1 TO 6), a(1 TO 4, 1 TO 2), b$(1 TO 4)
SCREEN 0
CLS 0
DEF fnx (c, r, x) = c + r * SIN(x)
DEF fny (c, r, x) = c - r * COS(x)
DEF fnd (d) = d / 100 * 3.14159
DEF fna (c, r, x) = c + r * COS(x + 3.14159 / 2)
DEF fnb (c, r, x) = c - r * SIN(x + 3.14159 / 2)
b$(1) = CHR$(&HFF)
b$(2) = CHR$(&H11)
b$(3) = CHR$(&H22) + CHR$(&H11)
b$(4) = CHR$(&HC) + CHR$(&HAA)
FOR i = 1 TO 6
    READ a$(i)
NEXT i
FOR i = 1 TO 4
    READ a(i, 1), a(i, 2)
NEXT i
CALL aa
CALL bb
CALL cc
CALL dd
END
DATA エイゴ, コゴ, シャカイ, スガク, ダンシ, ジョシ
DATA 85, 70, 75, 85, 80, 70, 88, 75

SUB aa
    LOCATE 1, 22: PRINT "ダンジョ ベツ セイキ"
    FOR i = 128 TO 224 STEP 32
        LINE (400, i)-(440, i + 16), , B
        PAINT (420, i + 8), b$((i - 96) / 32)
    NEXT i
    FOR i = 1 TO 4
        LOCATE i * 2 + 7, 60: PRINT a$(i)
    NEXT i
    LOCATE 19, 28: PRINT "100 テン"
    LOCATE 3, 20: PRINT "ジョシ"
END SUB

SUB bb
    CIRCLE (230, 184), 100
    LINE (230, 84)-(230, 284)

```

円の右側の点の X 座標値
 円の右側の点の Y 座標値
 成績を示す角度
 円の左側の点の X 座標値
 円の右側の点の Y 座標値
 ハッチング
 データの読み込み、課目
 データの読み込み、点数
 a(i,1)が男子、a(i,2)が女子
 グラフ準備ルーチンへ分岐
 男子の表示
 女子の表示
 目盛り線の表示
 グラフ準備ルーチン
 標題
 凡例
 凡例の課目名
 単位、項目名
 ダンシ

グラフ作成①
 外周の円
 基準線

```

FOR i = 1 TO 4 ————— 男子の 4 課目の点を示す帯
  PSET (fnx(230, 100 - (i - 1) * 16, 0), fny(184, 100 - (i - 1) * 16, 0))
  FOR q = 0 TO fnd(a(i, 1)) + .03 STEP .05 ————— 4 科目の外側の円弧
    x1 = fnx(230, 100 - (i - 1) * 16, q)
    y1 = fny(184, 100 - (i - 1) * 16, q)
    LINE -(x1, y1)
    xx1 = x1: yy1 = y1
  NEXT q
  PSET (fnx(230, 100 - i * 16, 0), fny(184, 100 - i * 16, 0)) ————— 4 科目の内側の円弧
  FOR q = 0 TO fnd(a(i, 1)) + .03 STEP .05
    x2 = fnx(230, 100 - i * 16, q)
    y2 = fny(184, 100 - i * 16, q)
    LINE -(x2, y2)
    xx2 = x2: yy2 = y2
  NEXT q
  LINE (xx1, yy1)-(xx2, yy2) ————— 帯の終端の線
  x3 = fnx(230, 100 - (i - 1) * 16 - 8, fnd(a(i, 1) / 2)) ————— ハッチング
  y3 = fny(184, 100 - (i - 1) * 16 - 8, fnd(a(i, 1) / 2))
  PAINT (x3, y3), b$(i)
NEXT i
END SUB

```

```

SUB cc ————— グラフ作成②
FOR i = 1 TO 4 ————— 女子の 4 課目の点を示す帯
  PSET (fna(230, 100 - (i - 1) * 16, 0), fnb(184, 100 - (i - 1) * 16, 0))
  FOR q = 0 TO fnd(a(i, 2)) + .03 STEP .05 ————— 4 課目の外側の円弧
    x1 = fna(230, 100 - (i - 1) * 16, q)
    y1 = fnb(184, 100 - (i - 1) * 16, q)
    LINE -(x1, y1)
    xx1 = x1: yy1 = y1
  NEXT q
  PSET (fna(230, 100 - i * 16, 0), fnb(184, 100 - i * 16, 0)) ————— 4 課目の内側の円弧
  FOR q = 0 TO fnd(a(i, 2)) + .03 STEP .05
    x2 = fna(230, 100 - i * 16, q)
    y2 = fnb(184, 100 - i * 16, q)
    LINE -(x2, y2)
    xx2 = x2: yy2 = y2
  NEXT q
  LINE (xx1, yy1)-(xx2, yy2) ————— 帯の終端の線
  x3 = fna(230, 100 - (i - 1) * 16 - 8, fnd(a(i, 2) / 2)) ————— ハッチング
  y3 = fnb(184, 100 - (i - 1) * 16 - 8, fnd(a(i, 2) / 2))
  PAINT (x3, y3), b$(i)
NEXT i
END SUB

```

```

SUB dd ————— 目盛り線と目盛り
FOR i = 70 TO 90 STEP 10 ————— 男子側の目盛り線
  px = fnx(230, 100, fnd(i))
  py = fnb(184, 100, fnd(i))
  LINE (230, 184)-(px, py)
NEXT i

```



```

FOR i = 70 TO 90 STEP 10
    px = fna(230, 100, fnd(i))
    py = fnb(184, 100, fnd(i))
    LINE (230, 184)-(px, py)
NEXT i
LOCATE 16, 16: PRINT "70"
LOCATE 16, 42: PRINT "70"
END SUB

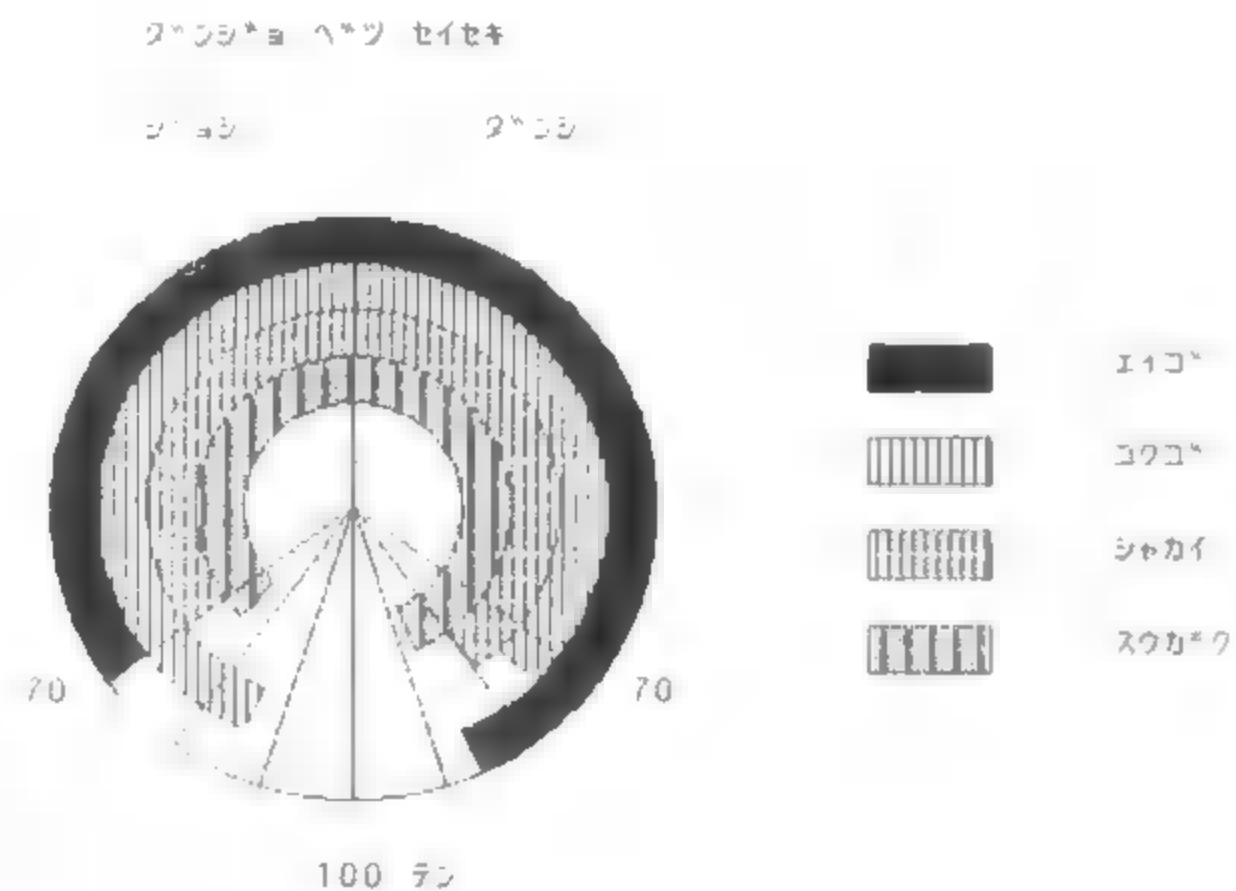
```

女子側の目盛り線

目盛り数値

目盛り数値

【結果】



【応用問題8】 虹グラフ

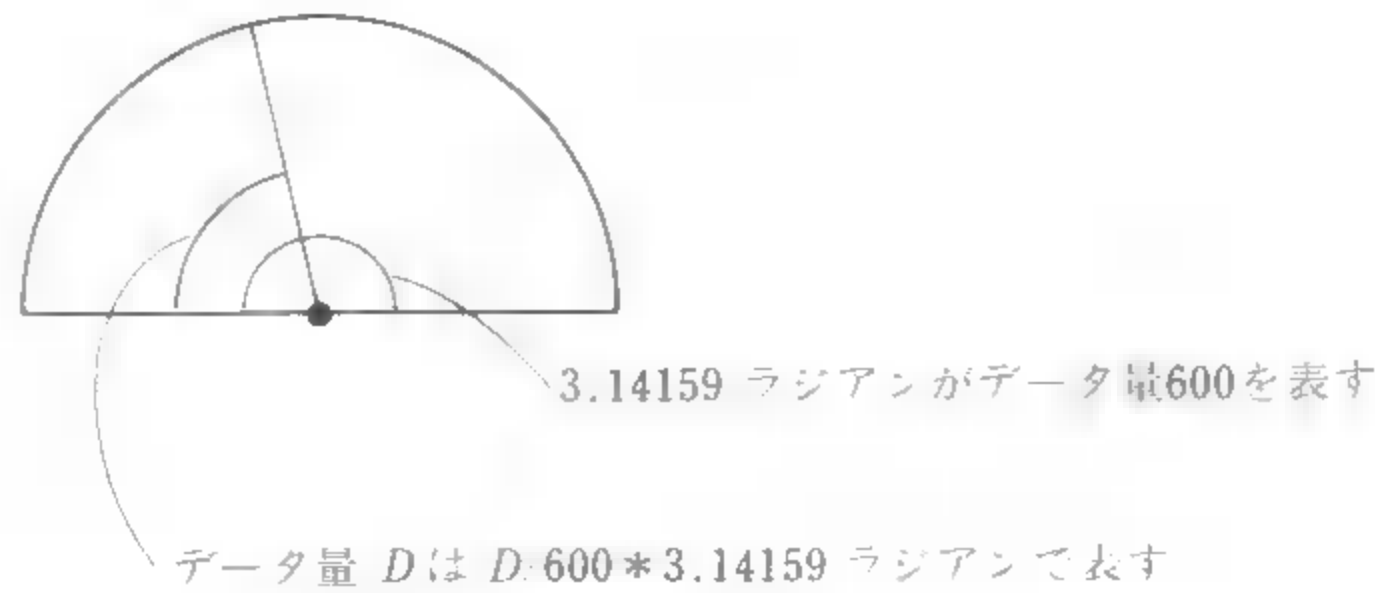
次表を虹グラフとするプログラムをつくれ。

商品別売上高 (1988年)

商 品	売上高 (万円)
クーラー	250
モーター	220
VTR	540
ステレオ	320
カメラ	100

【解 説】 ◎ 虹グラフをつくります。

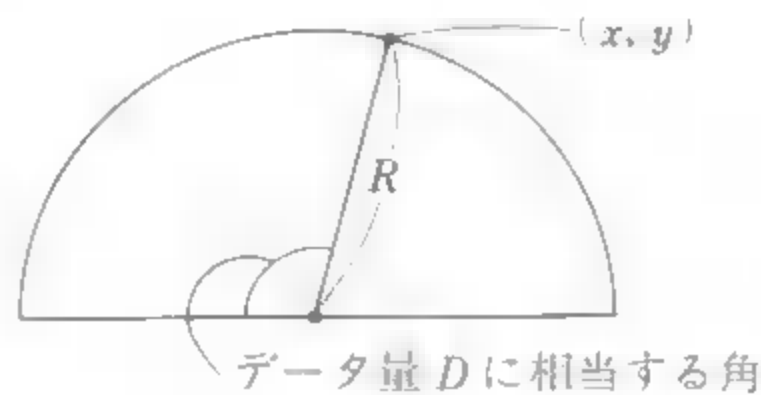
- ① 結果のように帯グラフの応用で左端を基準として、半円弧の右端までをデータ量の100%とする帯を外から内へ何重かに表わしたものを虹グラフと名づけます。
- ② 考え方の基本は【応用問題6】以降の帯グラフと同じです。基準が【応用問題6】では12時、データ量の100%を全周で表わしているのに対し、本グラフでは基準を9時の位置、データ量の100%を時計方向半周で表わします。
- ③ この場合のデータ量の最大を600とすると、データ量のDの角度は $D/600 \times 3.14159$ ラジアンとなります。



- ④ データ量 D の角を X とすると X , Y 座標は半径を R とすれば次のようになります.

$$X = R * \sin(X + 3.14159/2)$$

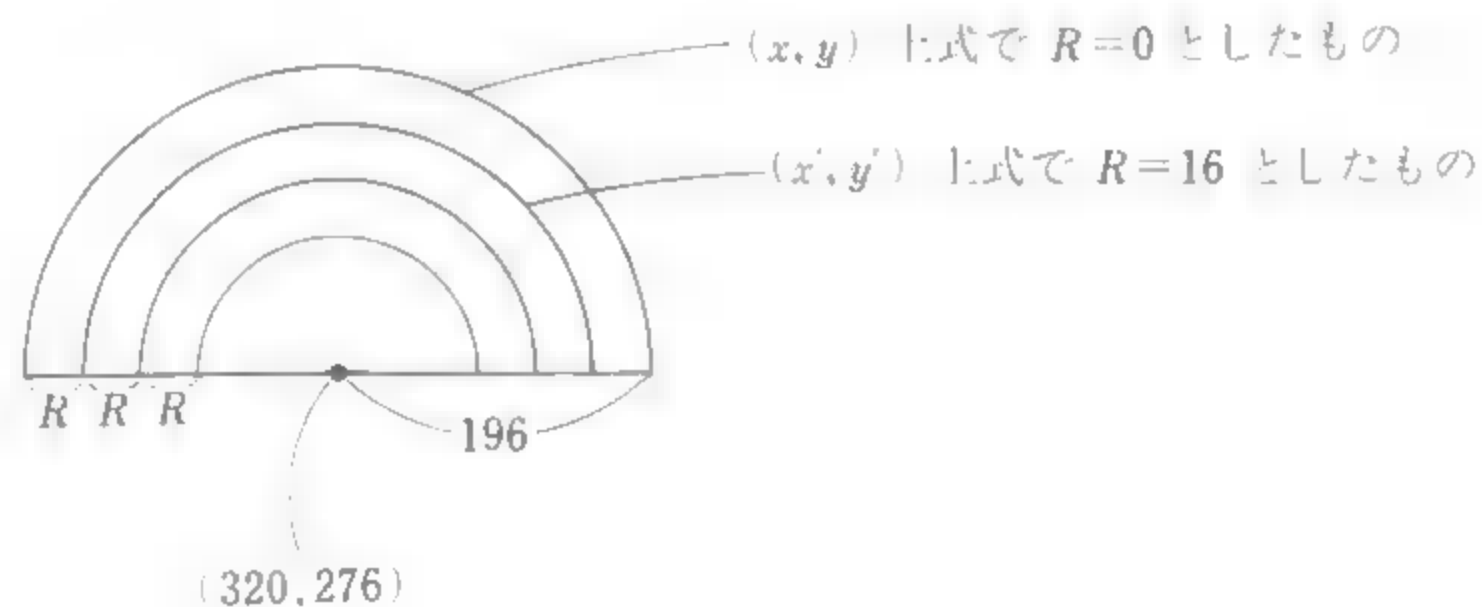
$$Y = R * \cos(X + 3.14159/2)$$



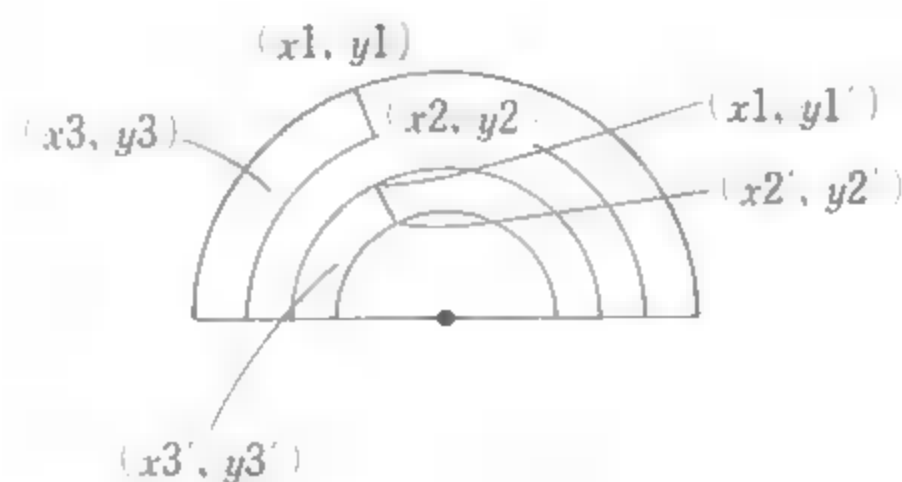
- ⑤ 中心を $(320, 276)$ とし, 一番外側の半円の半径を 196 とし, 内側の半円との間隔を R で与えなおすと

$$X = 320 - (196 + R) * \sin(X + 3.14159/2)$$

$$Y = 276 + (196 + R) * \cos(X + 3.14159/2) \text{ となります.}$$



- ⑥ 具体的には次のような手順で表示します.



- ① クーラーの値 250 に相当する角の点 $X1, Y1$ を求めます. 半径は 196 です.
- ② クーラーの値 250 に相当する角の点 $X2, Y2$ を求めます. 半径は $196 - 16$ です.
- ③ $(X1, Y1)$ と $(X2, Y2)$ を結びます.
- ④ 中間点 $(X3, Y3)$ を求めます.
- ⑤ $(X3, Y3)$ を指定して PAINT します.
- ⑥ ①～⑤をモーターの値 220 について行います. ①に相当する R は $16 * 2$ で, 半径は $196 - 32$ です. ②に相当する R は $16 * 3$ で 48, 半径は $196 - 48$ です.
- ⑦ 以下同様に VTR, ステレオ, カメラについて行います.

【プログラム例】

```

DECLARE SUB aa ()
DECLARE SUB bb ()
COMMON SHARED a$(), a(), b$()
DIM a$(1 TO 5), a(1 TO 5), b$(1 TO 5)
SCREEN 0
CLS 0
DEF fnx (r, x) = 320 - (196 + r) * SIN(x + 3.14159 / 2)
DEF fny (r, y) = 276 + (196 + r) * COS(y + 3.14159 / 2)
DEF fnd (d) = d / 600 * 3.14159
b$(1) = CHR$(&HFF)
b$(2) = CHR$(&H11)
b$(3) = CHR$(&H22) + CHR$(&H11)
b$(4) = CHR$(&HC) + CHR$(&HAA)
b$(5) = CHR$(&HC) + CHR$(&HA)
FOR i = 1 TO 5
    READ a$(i), a(i)
NEXT i
CALL aa
CALL bb
END
DATA クラリ , 250, 1171 , 220, VTR , 540
DATA ステレオ , 320, カメラ , 100

SUB aa
LOCATE 1, 28: PRINT "ショウヒンヘツ クリゲダカ(1987ネン)"
FOR i = 0 TO 9
    LINE (128 + i * 16 - 4, 276)-(128 + i * 16 - 4, 376)
    CIRCLE (320, 276), 196 - 16 * i, , 0, 3.14159
    LINE (520 - i * 16 - 4, 276)-(520 - i * 16 - 4, 376)
NEXT i
LINE (116, 276)-(268, 276)
LINE (524, 276)-(372, 276)
LINE (124, 376)-(268, 376)
LINE (516, 376)-(372, 376)
FOR i = 1 TO 5
    LOCATE 19, 13 + 4 * i: PRINT MIDS(a$(i), 1, 1)
    LOCATE 20, 13 + 4 * i: PRINT MIDS(a$(i), 2, 1)
    LOCATE 21, 13 + 4 * i: PRINT MIDS(a$(i), 3, 1)
    LOCATE 22, 13 + 4 * i: PRINT MIDS(a$(i), 4, 1)
    LOCATE 19, 44 + 4 * i: PRINT MIDS(a$(6 - i), 1, 1)
    LOCATE 20, 44 + 4 * i: PRINT MIDS(a$(6 - i), 2, 1)
    LOCATE 21, 44 + 4 * i: PRINT MIDS(a$(6 - i), 3, 1)
    LOCATE 22, 44 + 4 * i: PRINT MIDS(a$(6 - i), 4, 1)
    LOCATE 18, 13: PRINT "0"
    LOCATE 11, 15: PRINT "100"
    LOCATE 6, 25: PRINT "200"
    LOCATE 4, 40: PRINT "300"
    LOCATE 6, 54: PRINT "400"
    LOCATE 11, 64: PRINT "500"
    LOCATE 18, 68: PRINT "600マリン"
    ii = i * 100
    LINE (fnx(8, fnd(ii)), fny(8, fnd(ii)))-(fnx(0, fnd(ii)), fny(0, fnd(ii)))
NEXT i
END SUB
SUB bb

```

データ量に相当する半円周上の
点の x, y 座標値

データ量に相当する角度

ハッチングパターンの登録

データの読み込み

グラフ準備ルーチンへ分岐

グラフ作成ルーチンへ分岐

グラフ準備

標題

直線+円弧の表示

部品名をかく部分の横線

商品名

目盛り数値と単位

目盛り線

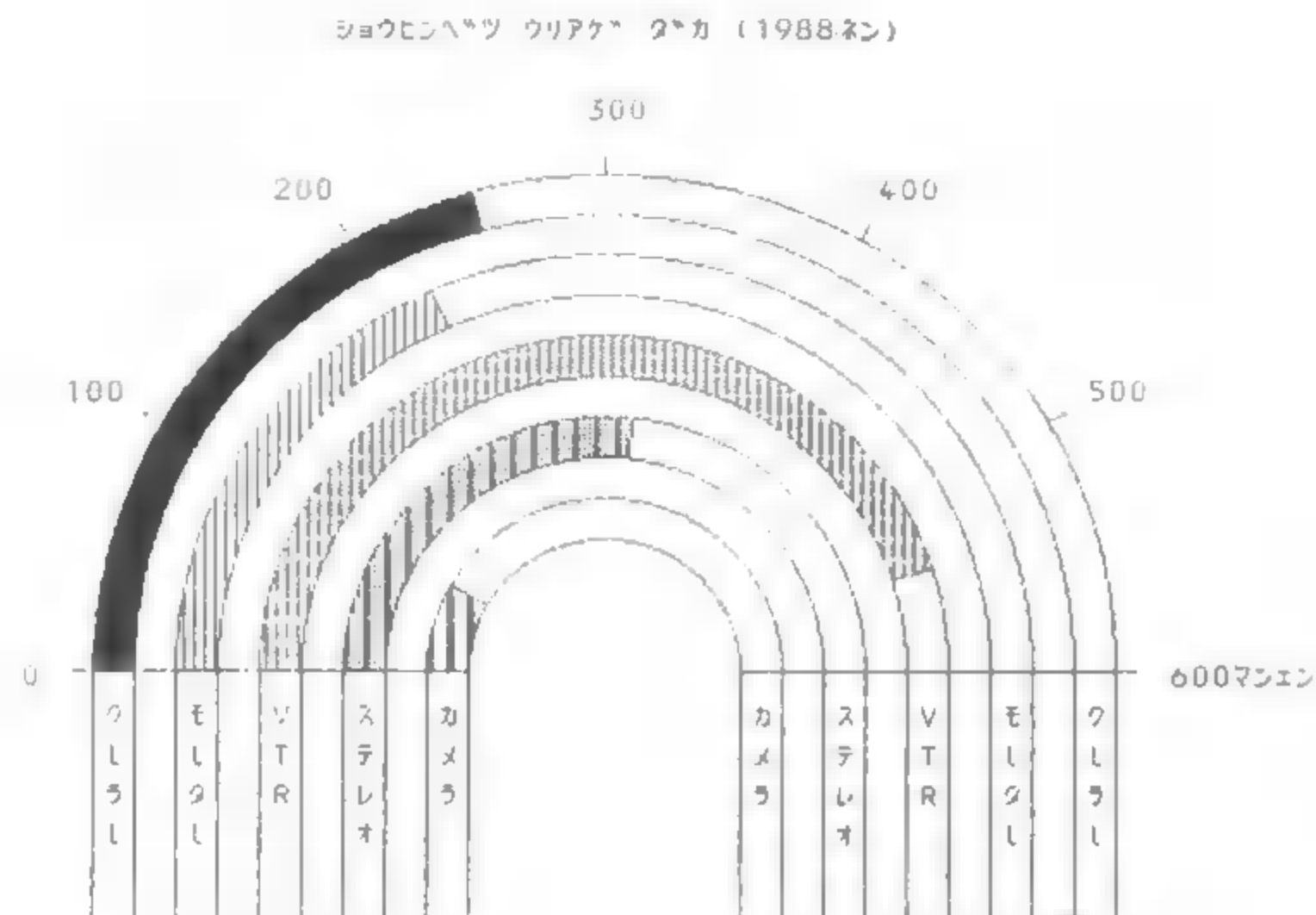
帯表示

```

FOR i = 1 TO 5 ————— 帯の終端部の表示
  ii = (4 - 4 * i) * 8
  iii = ii - 16
  LINE (fnx(ii, fnd(a(i))), fny(ii, fnd(a(i))))-(fnx(iii, fnd(a(i))), fny(iii, fnd(a(i)))) — 区切り線
  PAINT (fnx((ii - 4), fnd(a(i) / 2)), fny((ii - 4), fnd(a(i) / 2))), b$(i) — ハッチング
NEXT i
END SUB

```

【結 果】



【応用問題 9】 放射線グラフ

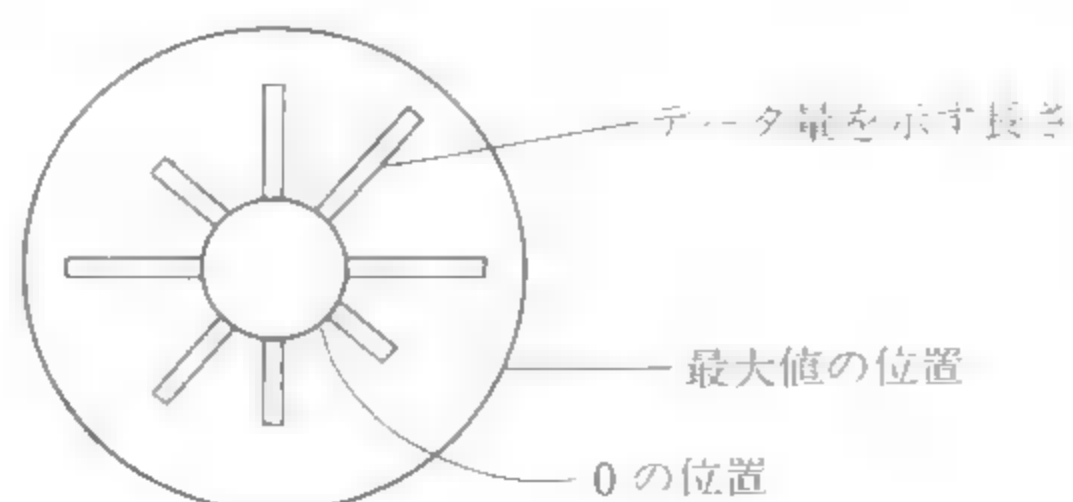
下の表を放射線グラフにするプログラムをつくれ。

交通量調査 (1989年10月10日)

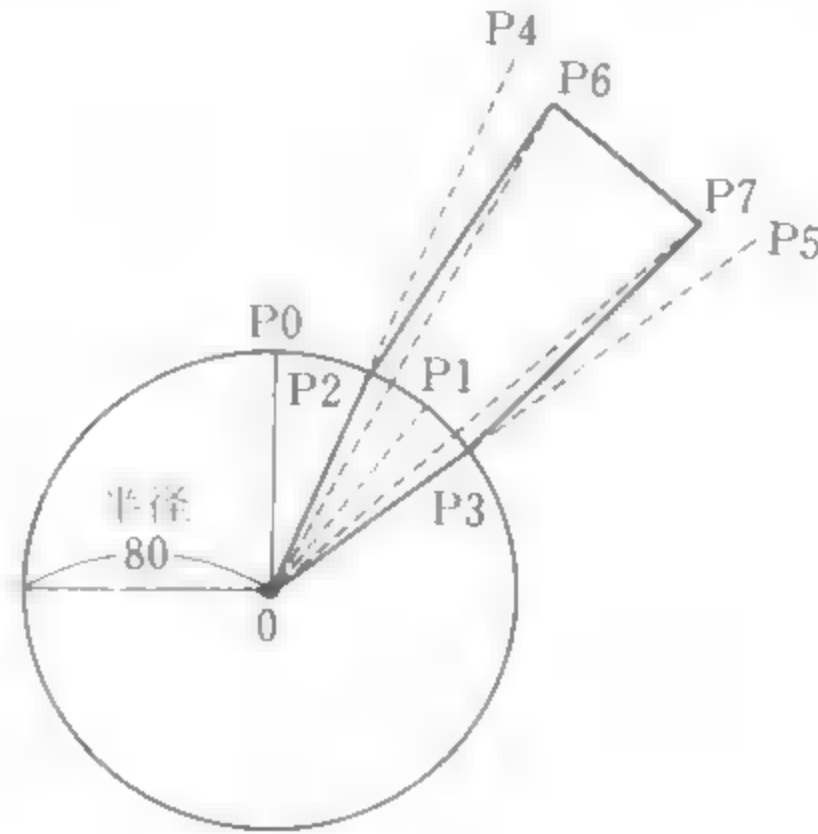
時間	1000台	時間	1000台	時間	1000台	時間	1000台
1	100	7	250	13	180	19	490
2	50	8	500	14	190	20	420
3	60	9	400	15	250	21	320
4	90	10	350	16	400	22	250
5	120	11	300	17	470	23	200
6	180	12	250	18	500	24	100

【解 説】 ◎ 放射グラフをつくります。

- ① 中心からデータ量に応じた長さの棒を放射線状にかくグラフを放射線グラフと名づけます。



- ② 中心を(300,200)とし、半径を80の円をかきます。これが真中の円で統計量0の基準となります。
- ③ ここでは1時から24時まで24個の時系列データを時計まわりで表示します。
- ④ 1時のデータを100とすると次のような手順で表示します。



- ① 全体で24本の棒を表示しますから、1時の位置角P0-O-P1は $1/24 \times 2 \times 3.14159$ ラジアンです。P0-O-P1の角を中心にして棒の幅をもたせるため-0.8ラジアンと+0.8ラジアンの角を考えます。中心(300,200)、半径80で、角を $1/24 \times 2 \times 3.14159 - 0.8$ ラジアンの点がP2、角を $1/24 \times 2 \times 3.14159 + 0.8$ ラジアンの点がP3です。P2が棒のかき始めの点、P3が棒のかきおえた点です。
- ② 次に棒の長さは半径20ドットについて100台としますので、データ量/100*20が棒の長さです。中心からの長さ(半径)ではそれに80を加えたものです。
- ③ したがって、中心を(300,200)、半径を $80 + 100/20$ 角度を $1/24 \times 2 \times 3.14159 - 0.8$ ラジアンの点P4と、角度を $1/24 \times 2 \times 3.14159 + 0.8$ ラジアンの点P5を求めP2-P4-P5-P3と結ぶと棒になります。しかし、棒の長さが長くなればなるほど、先が開く扇形になりますので、-I大します。
- ④ P0-O-P1から-0.6ラジアン、+0.6ラジアンの角を先端では使います。半径(300,200)、半径 $80 + 100/100 \times 20$ 角度を $1/24 \times 2 \times 3.14159 - 0.6$ ラジアンの点をP6、角度を $1/24 \times 2 \times 3.14159 + 0.6$ ラジアンの点をP7とします。P2-P6-P7-P3を結びます。棒となります。

【プログラム例】

```

DECLARE SUB aa ()
DECLARE SUB bb ()
COMMON SHARED a(), b$
DIM a(1 TO 24)
SCREEN 0
CLS 0
DEF fnx (c, r, x) = c + r * SIN(x)
DEF fny (c, r, x) = c - r * COS(x)
DEF fnd (d, e) = d / 24 * 2 * 3.14159 + e
DEF fne (f) = f / 100 * 20 + 80
b$ = CHR$(&H22) + CHR$(&H11)
FOR i = 1 TO 24
    READ a(i)
NEXT i
CALL aa
CALL bb
END
DATA 100, 50, 60, 90, 120, 180, 250, 500, 400, 350, 300, 250
DATA 180, 190, 250, 400, 470, 500, 490, 420, 320, 250, 200, 100

```

中心の x 座標を c、半径を r、角を x とする円周上の点の x 座標値
 中心の Y 座標値を c、半径を r、角を x とする円周上の点の Y 座標値
 D 時 (D は 1 から 24) のときの角を中心として +e、-e ラジアン
 データ量 F に相当する棒の長さ +80 で F に相当する半径を示す
 ハッチング
 データの読み込み
 グラフ準備ルーチンへ分岐
 グラフ作成ルーチンへ分岐

```

SUB aa ----- グラフ準備ルーチン
LOCATE 1, 26: PRINT "コウツクリョウ チョウサ (1989 ネン10ガ ツ10カ)" ----- 標題
LOCATE 9, 37: PRINT 0 ----- 時間と目盛り数値, 単位
LOCATE 10, 38: PRINT "ジ"
LOCATE 13, 45: PRINT 6
LOCATE 16, 37: PRINT 1
LOCATE 17, 37: PRINT 2
LOCATE 13, 28: PRINT 18
LOCATE 13, 53: PRINT 2
LOCATE 13, 56: PRINT 3
LOCATE 13, 58: PRINT 4
LOCATE 13, 62: PRINT "5 (100ダイ)" -----
END SUB

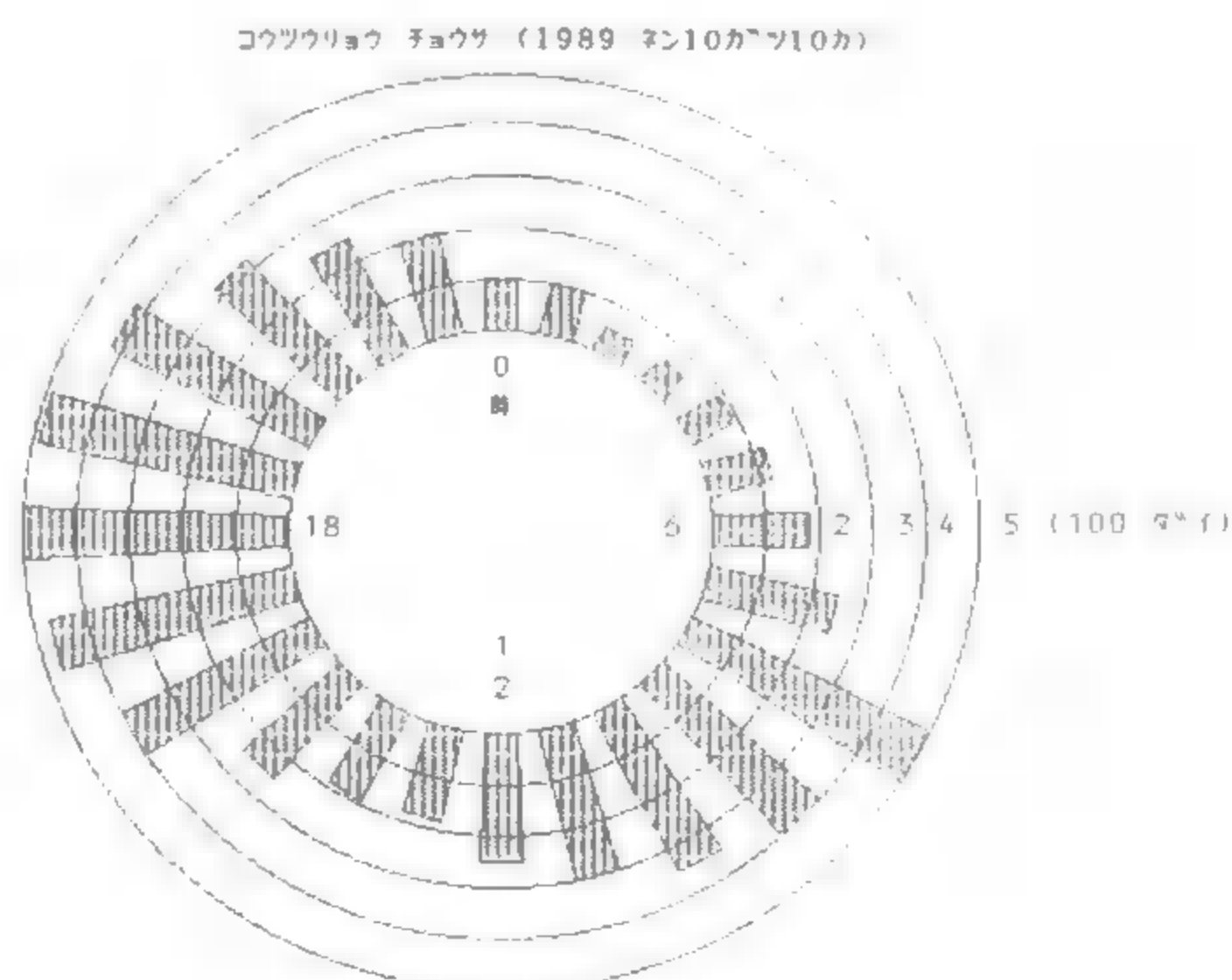
```

```

SUB bb ----- グラフ作成ルーチン
CIRCLE (300, 200), 80 ----- 円
FOR i = 1 TO 24 ----- 棒の作成
    PSET (fnx(300, fne(0), fnd(i, -.08)), fny(200, fne(0), fnd(i, -.08))) ----- 棒
    x1 = fnx(300, fne(a(i)), fnd(i, -.06))
    y1 = fny(200, fne(a(i)), fnd(i, -.06))
    x2 = fnx(300, fne(a(i)), fnd(i, .06))
    y2 = fny(200, fne(a(i)), fnd(i, .06))
    x3 = fnx(300, fne(0), fnd(i, .08))
    y3 = fny(200, fne(0), fnd(i, .08))
    LINE -(x1, y1)
    LINE -(x2, y2)
    LINE -(x3, y3)
    x4 = (x1 + x3) / 2 ----- ハッチング
    y4 = (y1 + y3) / 2
    PAINT (x4, y4), b$ -----
NEXT i
FOR i = 100 TO 180 STEP 20 ----- 同心円
    CIRCLE (300, 200), i
NEXT i
END SUB

```

【結 果】



【応用問題 10】 レーダーチャート

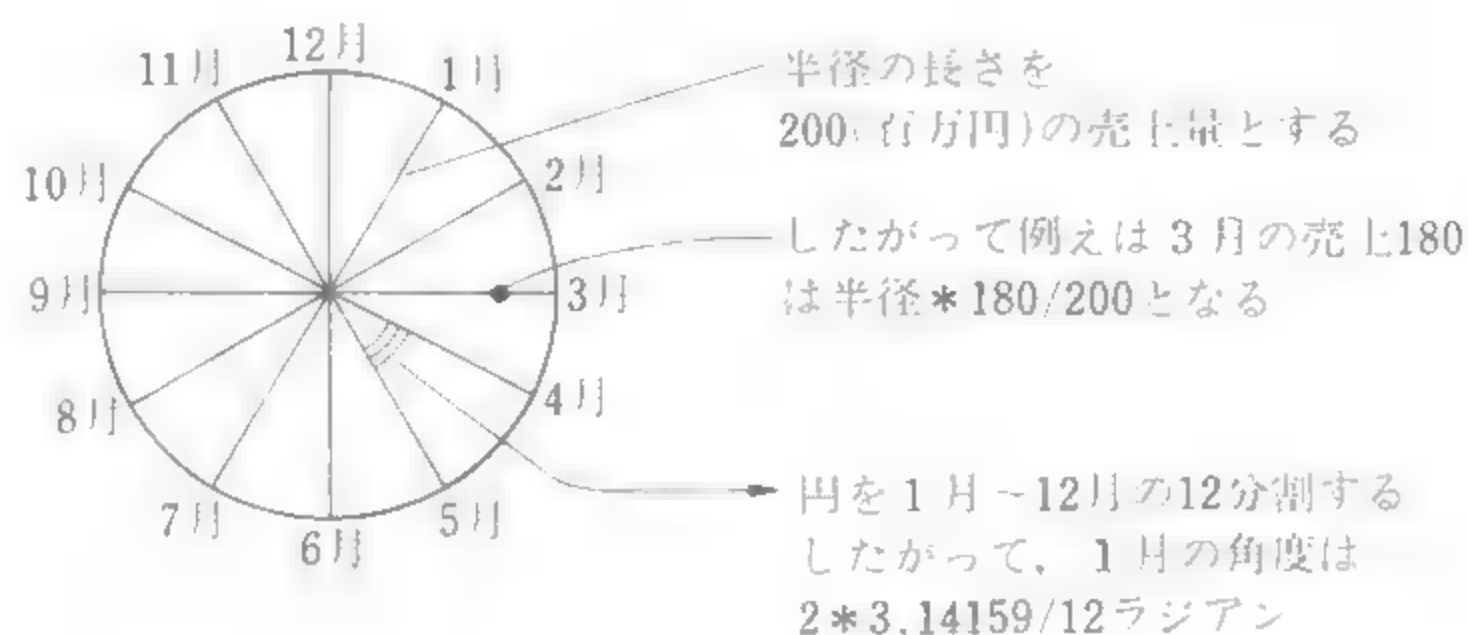
下表をレーダーチャートにかくプログラムをつくれ。

月別売上

月	売上(100万円)	月	売上(100万円)
1	80	7	140
2	60	8	90
3	180	9	150
4	160	10	160
5	170	11	130
6	120	12	120

【解 説】 ① レーダーチャートをつくります。

① 下図のような考えでレーダーチャートをつくります。



したがって、中心を C 、データ量を D 、円の半径を 200 とすると M 月の値のドットの表示位置は
 $X = C + 200 * 180 / 200 * \sin(M * 2 * 3.14159 / 12)$
 $Y = C - 200 * 180 / 200 * \cos(M * 2 * 3.14159 / 12)$ となります。

【プログラム例】

```

DECLARE SUB aa ()
DECLARE SUB bb ()
COMMON SHARED a()
DIM a(1 TO 12)
SCREEN 0
CLS 0
DEF fnx (c, r, x) = c + r * SIN(x / 12 * 2 * 3.14159)
DEF fny (c, r, x) = c - r * COS(x / 12 * 2 * 3.14159)
DEF fnd (d) = d / 200 * 100
FOR i = 1 TO 12
    READ a(i)
NEXT i
CALL aa
CALL bb
END
DATA 80, 60, 180, 160, 170, 120
DATA 140, 90, 150, 160, 130, 120

```

データ量をプロット
 X 、 Y 座標値
データ量に相当する半径
データの読み込み
円の半径が 100
グラフ作成準備ルーチンへ分岐
グラフ作成ルーチンへ分岐


```

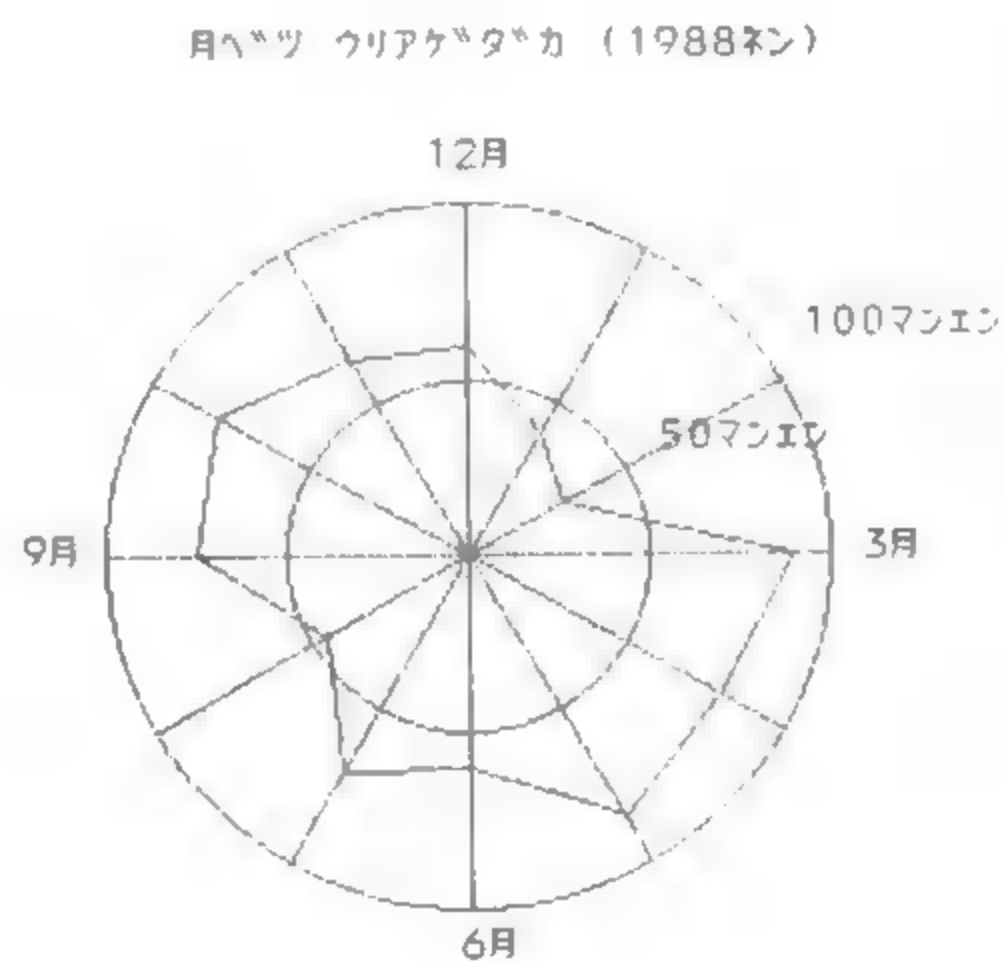
SUB aa ----- グラフ作成準備ルーチン
LOCATE 4, 29: PRINT "ツキヘッ ウリアケタカ (1988 ネン)" ----- 標題
LOCATE 6, 37: PRINT "12ガツ" ----- 項目名, 単位
LOCATE 13, 52: PRINT "3ガツ"
LOCATE 20, 38: PRINT "6ガツ"
LOCATE 13, 21: PRINT "9ガツ"
LOCATE 9, 50: PRINT "100マンエン"
LOCATE 11, 45: PRINT "50マンエン"
CIRCLE (300, 200), 100 ----- 円と目盛り線の表示
CIRCLE (300, 200), 50
FOR i = 1 TO 12
  x = fnx(300, 100, i)
  y = fny(200, 100, i)
  LINE (300, 200)-(x, y)
NEXT i
END SUB

SUB bb ----- グラフ作成ルーチン
PSET (fnx(300, fnd(a(12)), 0), fny(200, fnd(a(12)), 0)) ----- プロット
FOR i = 1 TO 12
  x = fnx(300, fnd(a(i)), i)
  y = fny(200, fnd(a(i)), i)
  LINE -(x, y)
NEXT i
END SUB

```

12月-1月
-2月……
-12月のデータを順にむすぶ

【結 果】



【応用問題 11】 絵画グラフ

下の業種別景気状況を、景気の状態ごとの顔色で表わす絵画グラフをつくれ。なお、景気の状態は①超繁忙 ②好景気 ③普通 ④不調 ⑤不景気とし、おのこの顔色は、結果のようなマークとする。

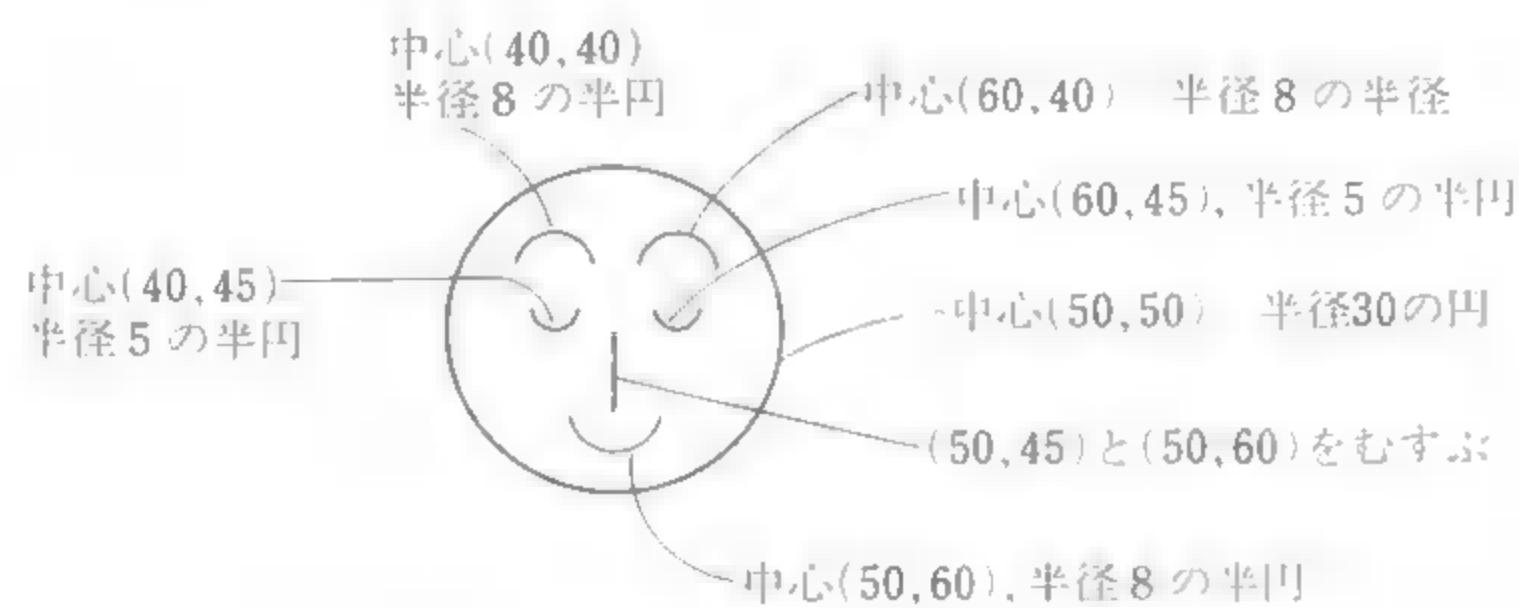
業種別景気予想

業 種	景気予想
化学工業	好 景 気
鉄 鋼 業	普 通
繊維工業	不 況
電機工業	不 調
機械工業	超 繁 忙
商 業	不 況

【解 説】 ◎ 絵画グラフをつくります。

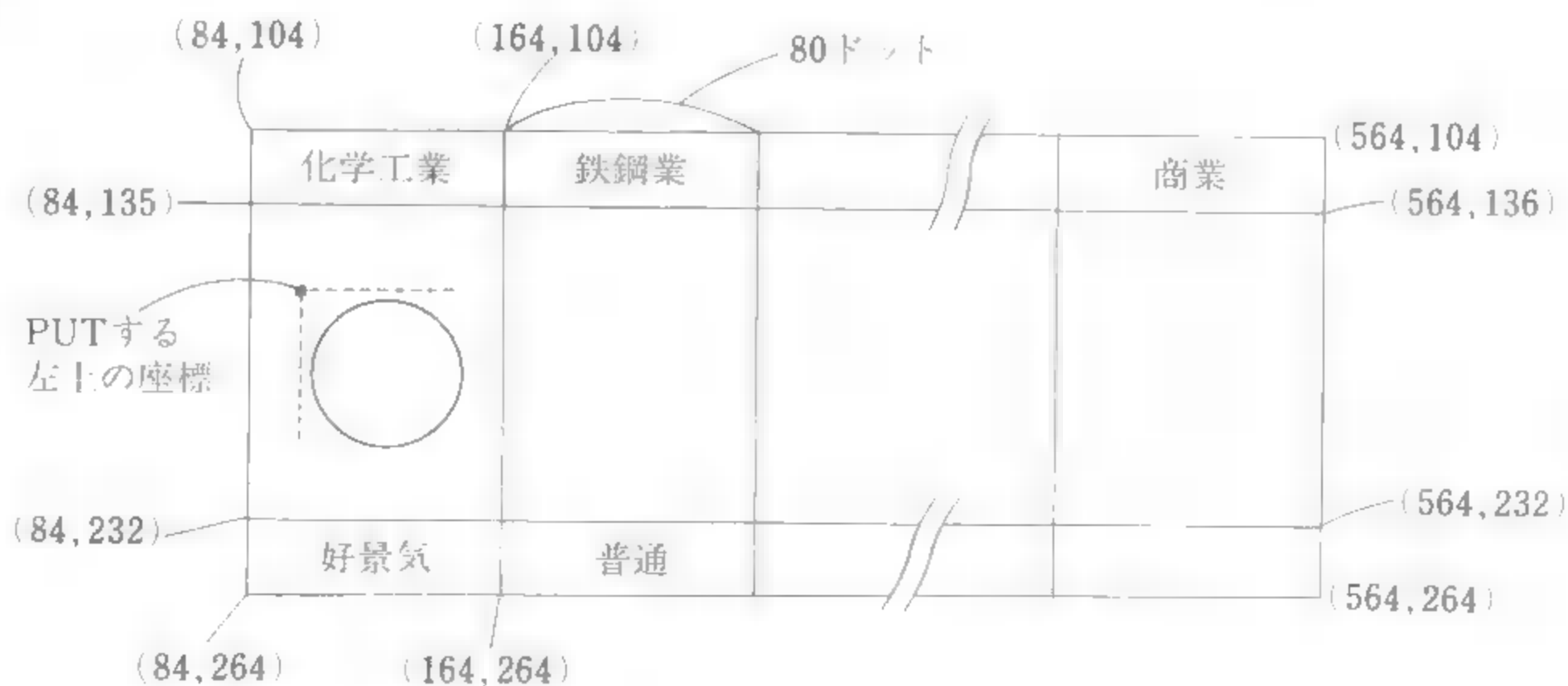
- ① 景気の状態による顔色を 5 種用意します。おのこの、グラフィックで表示した後配列 A%~E%に GETa でとりこみます。

好景気の例を示します。



左上 (20,20) と右下 (80,80) を指定して配列 B%に取り込む

- ② 表の枠、文字を表示し、絵画の部分に顔色が格納されている配列を PUT で表示します。



【プログラム例】

```

DECLARE SUB aa ()
DECLARE SUB bb ()
DECLARE SUB cc ()
DECLARE SUB dd ()
DECLARE SUB ee ()
DECLARE SUB ff ()
DECLARE SUB gg ()
COMMON SHARED a%(), b%(), c%(), d%(), e%(), a$(), b$()——— 5 種の顔の入る配列
DIM a%(1000), b%(1000), c%(1000), d%(1000), e%(1000)
DIM a$(1 TO 6, 1 TO 2), b$(1 TO 5)
SCREEN 0
CLS 0
FOR i = 1 TO 6
    READ a$(i, 1), a$(i, 2)
NEXT i
FOR i = 1 TO 5
    READ b$(i)
NEXT i
CALL aa
CALL bb
CALL cc
CALL dd
CALL ee
CLS 0
CALL ff
CALL gg
END
DATA 化学工業, 2, 鉄鋼業, 3, 繊維工業, 5
DATA 電機工業, 4, 機械工業, 1, 商業, 5
DATA 超繁忙, 好景気, 普通, 不調, 不況

```

```

SUB aa
    CLS 0
    CIRCLE (50, 50), 30
    CIRCLE (40, 40), 8, , 0, 3.14159
    CIRCLE (60, 40), 8, , 0, 3.14159
    CIRCLE (40, 45), 5, , 0, 3.14159
    CIRCLE (60, 45), 5, , 0, 3.14159
    LINE (50, 45)-(50, 60)
    CIRCLE (50, 60), 8, , 3.14159, 2 * 3.14159
    GET (20, 20)-(80, 80), a%
END SUB

```

```

SUB bb
    CLS 0
    CIRCLE (50, 50), 30
    CIRCLE (40, 40), 8, , 0, 3.14159
    CIRCLE (60, 40), 8, , 0, 3.14159
    CIRCLE (40, 45), 5, , 3.14159, 2 * 3.14159
    CIRCLE (60, 45), 5, , 3.14159, 2 * 3.14159

```

```

LINE (50, 45)-(50, 60)
CIRCLE (50, 60), 8, , 3.14159, 2 * 3.14159
GET (20, 20)-(80, 80), b%
END SUB

```

```

SUB cc ----- 顔色 3 の配列 c% への取り込み
CLS 0
CIRCLE (50, 50), 30
CIRCLE (40, 40), 8, , 0, 3.14159
CIRCLE (60, 40), 8, , 0, 3.14159
CIRCLE (40, 45), 5, , 3.14159, 2 * 3.14159
CIRCLE (60, 45), 5, , 3.14159, 2 * 3.14159
LINE (50, 45)-(50, 60)
LINE (40, 65)-(60, 65)
GET (20, 20)-(80, 80), c%
END SUB

```

```

SUB dd ----- 顔色 4 の配列 d% への取り込み
CLS 0
CIRCLE (50, 50), 30
LINE (32, 35)-(48, 35)
LINE (52, 35)-(68, 35)
LINE (34, 45)-(46, 45)
LINE (54, 45)-(66, 45)
LINE (50, 47)-(50, 60)
LINE (40, 65)-(60, 65)
GET (20, 20)-(80, 80), d%
END SUB

```

```

SUB ee ----- 顔色 5 の配列 e% への取り込み
CLS 0
CIRCLE (50, 50), 30
LINE (35, 30)-(45, 40)
LINE (55, 40)-(65, 30)
LINE (32, 55)-(45, 45)
LINE (55, 45)-(68, 55)
LINE (50, 50)-(50, 60)
LINE (40, 70)-(50, 65)
LINE -(60, 70)
GET (20, 20)-(80, 80), e%
END SUB

```

```

SUB ff ----- グラフ作成準備ルーチン
LOCATE 5, 33: PRINT "業種別景気予想" ----- 標題
FOR i = 84 TO 564 STEP 80 ----- 表のわく
    LINE (i, 104)-(i, 264)
NEXT i
LINE (84, 104)-(564, 104)
LINE (84, 135)-(564, 136)
LINE (84, 232)-(564, 232)
LINE (84, 264)-(564, 264)
END SUB

```

```
SUB gg
FOR i = 1 TO 6
LOCATE 8, 2 + i * 10: PRINT a$(i, 1)
x = VAL(a$(i, 2))
LOCATE 16, 4 + i * 10: PRINT b$(x)
IF a$(i, 2) = "1" THEN PUT (16 + 80 * i, 150), a%
IF a$(i, 2) = "2" THEN PUT (16 + 80 * i, 150), b%
IF a$(i, 2) = "3" THEN PUT (16 + 80 * i, 150), c%
IF a$(i, 2) = "4" THEN PUT (16 + 80 * i, 150), d%
IF a$(i, 2) = "5" THEN PUT (16 + 80 * i, 150), e%
NEXT i
END SUB
```

【結 果】

業種別景気予想

化学工業	鉄鋼業	繊維工業	電機工業	機械工業	商業
					
好景気	普通	不況	不調	超繁忙	不況

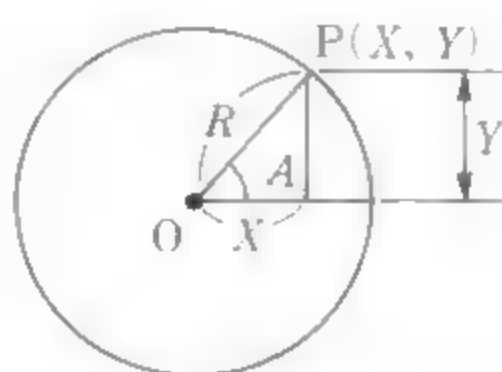
4章 媒介変数を使った図形

【応用問題 12】 円

円を表示するプログラムをつくれ。

【解 説】 ◎ 媒介変数を使って円を表示します。

- ① 円の表示は媒介変数を使うと便利ですが、この方法は一般によく使われる方法ですから少しでもグラフィックをした人はご存知でしょう。復習と媒介変数の意味を確認するためにあえて説明します。
- ② 下の図の円周上の点 P は次のように表わせます。



$$X = R \cos(A)$$

$$y = R \sin(A)$$

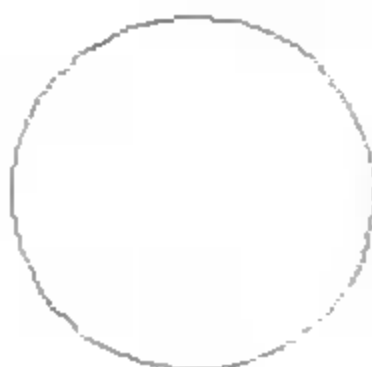
ただし、O を原点とする半径 R の円で X 軸と OP を結ぶ直線のつくる角を A とします。

- ③ このとき A の値を変えると (X, Y) の値は円周上の点 P' となります。
- ④ したがって、A を 0 度から 360 度まで、細かい間隔でたとえば 1 度ずつ増やしその点を PSET (表示) するか、1 つ後の点と順に結んでいけば円となります。
- ⑤ このように、A の値を変化させることで図形が描けます。この A を媒介変数と呼びます。本章では媒介変数を使っていろいろな図形を表示します。

【プログラム例】

```
SCREEN 0
CLS 0
DEF fnx (x) = 300 + 50 * COS(x) —— X の定義
DEF fny (x) = 200 - 50 * SIN(x) —— Y の定義
PSET (fnx(0), fny(0)) —— スタート点へポインタをおく
FOR i = 0 TO 2 * 3.14159 STEP .01 —— 作図
    LINE -(fnx(i), fny(i))
NEXT i
END
```

【結 果】



【応用問題 13】 三ツ葉

三ツ葉の曲線をかくプログラムをつくれ.

【解 説】

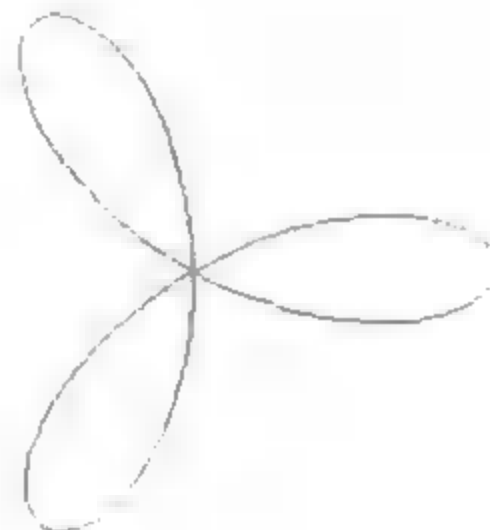
① $X=300+50\cos(2x)+50\cos(x)$

$Y=200-50\sin(2x)+50\sin(x)$ として x を 0 ラジアンから 2π ラジアンまで 0.1 ラジアン単位で増加させて, LINE 文で結びます.

② 結果のような花びら形となります.

【プログラム例】

```
SCREEN 0
CLS 0
DEF fnx (x) = 300 + 50 * COS(2 * x) + 50 * COS(x) —— 円周上の点の X 座標値
DEF fny (x) = 200 - 50 * SIN(2 * x) + 50 * SIN(x) —— 円周上の点の Y 座標値
PSET (fnx(0), fny(0)) —— 図の表示
FOR i = 0 TO 2 * 3.14159 + .1 STEP .1
    LINE -(fnx(i), fny(i))
NEXT i
END
```

【結 果】**【応用問題 14】 5 枚の花びらの形**

【応用問題 13】の 3 枚の花びらの形を変形し, 5 枚の花びらの形とするプログラムをつくれ.

【解 説】 ◎ 5 枚の花びらの形をつくります.

① 【応用問題 13】のプログラムを変形します.

② $X=300+50\cos(4x)+50\cos(x)$

$Y=200-50\sin(4x)+50\sin(x)$ とします.

基本形が x であった部分の一部を $4x$ としています.

③ この結果, 5 枚の花びらの形となります. $4x$ の $4+1$ が花びらの数です. したがって, ここを変化させれば, 何枚の花びらでもかけます.

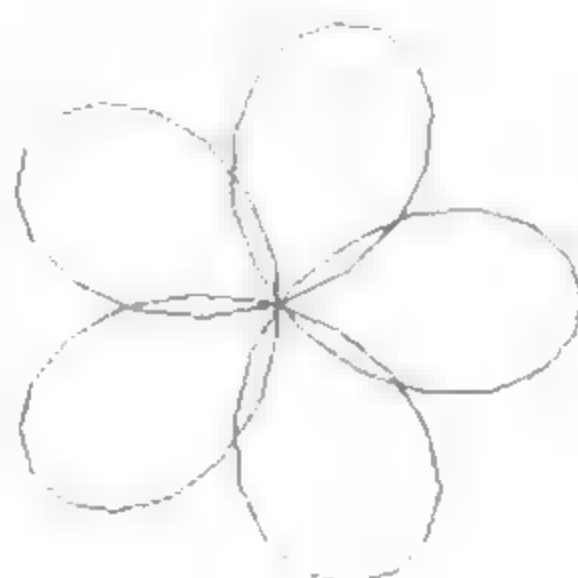
【プログラム例】

```

SCREEN 0
CLS 0
DEF fnx (x) = 300 + 50 * COS(4 * x) + 50 * COS(x) —— X 座標値
DEF fny (x) = 200 - 50 * SIN(4 * x) + 50 * SIN(x) —— Y 座標値
PSET (fnx(0), fny(0)) —— i を 0 から  $2\pi$  まで (fnx,fny)
FOR i = 0 TO 2 * 3.14159 + .1 STEP .1 —— 点をむすぶ
    LINE -(fnx(i), fny(i))
NEXT i
END

```

【結 果】



【応用問題 15】 三ツ葉の変形 1

【応用問題 13】の花びらの形を【結 果】のように変形させるプログラムをつくれ。

【解 説】

- ① 【応用問題 13】のプログラムを変形します。
- ② $X=300+50\cos(2x)+50\cos(x)$
 $Y=200-100\sin(2x)+100\sin(x)$ とします。
 基本形が 50 であった一部を 100 とします。
- ③ この結果、縦方向が長くなった形となります。これは Y の値（縦方向）を変えていますから、縦方向のみが変化するの当然です。
- ④ 100 の部分を大きくすると縦長、逆に小さくすると縦短となります。
- ⑤ 簡単に類推できるように x の値を変えれば横長、横短となります。
 $50\cos(2x)+50\cos(x)$ の 50 を変えて試みてください。

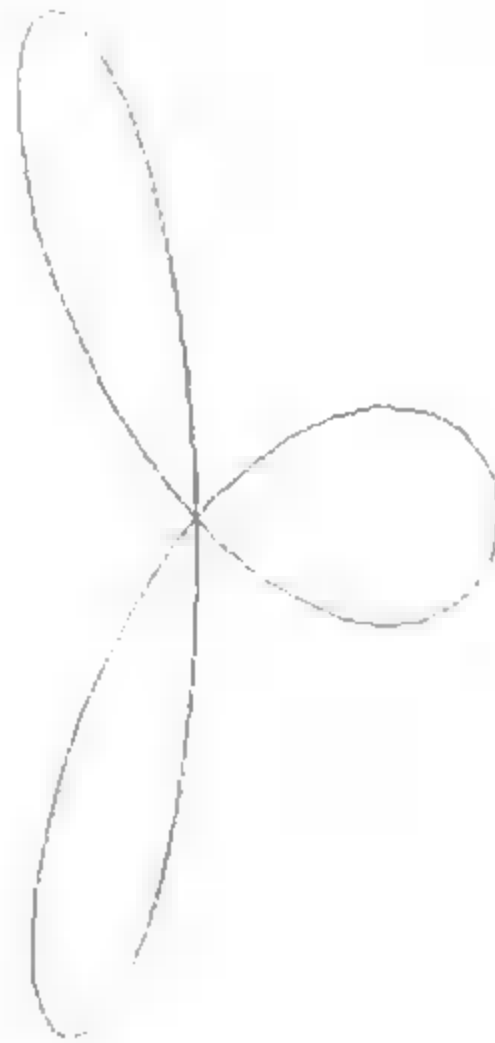
【プログラム例】

```

SCREEN 0
CLS 0
DEF fnx (x) = 300 + 50 * COS(2 * x) + 50 * COS(x) —— X 座標値
DEF fny (x) = 200 - 100 * SIN(2 * x) + 100 * SIN(x) —— Y 座標値
PSET (fnx(0), fny(0)) —— i を 0 から  $2\pi$  まで
FOR i = 0 TO 2 * 3.14159 + .1 STEP .1 —— (fnx,fny) 点をむすぶ
    LINE -(fnx(i), fny(i))
NEXT i
END

```

【結 果】



【応用問題 16】 三ツ葉

【応用問題 13】の花びらの形を【結 果】のように変形させるプログラムをつくれ.

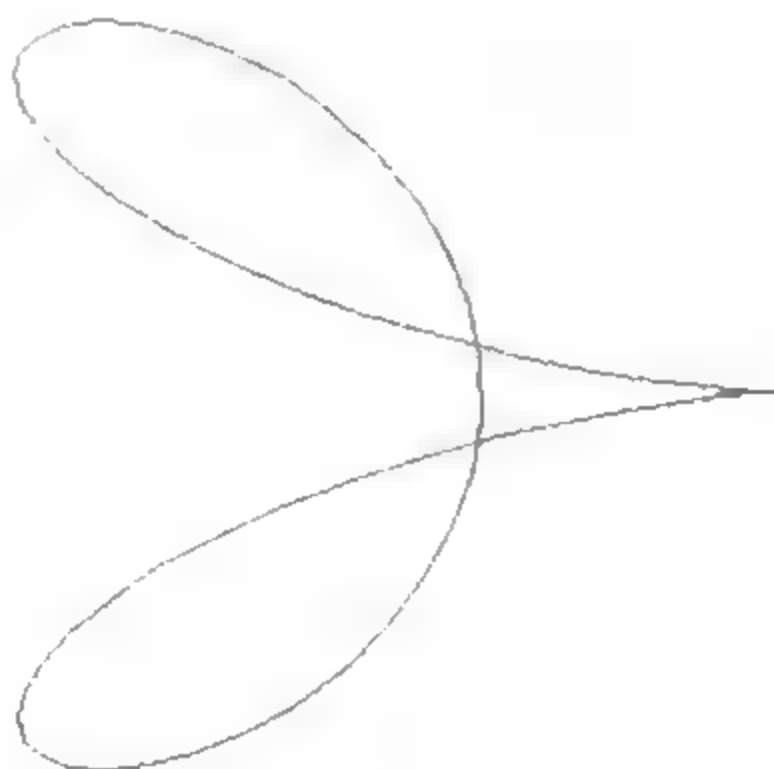
【解 説】

- ① 【応用問題 13】のプログラムを変形します.
- ② $X=300+100\cos(2x)+50\cos(x)$
 $Y=200-50\sin(2x)+100\sin(x)$ とします.
 基本形が 50 であった一部を 100 とします.
- ③ この結果, 【結 果】のような図となります.

【プログラム例】

```
SCREEN 0
CLS 0
DEF fnx (x) = 300 + 100 * COS(2 * x) + 50 * COS(x) —— X 座標値
DEF fny (x) = 200 - 50 * SIN(2 * x) + 100 * SIN(x) —— Y 座標値
PSET (fnx(0), fny(0)) —— i を 0 から 2π まで
FOR i = 0 TO 2 * 3.14159 + .1 STEP .1 —— (fnx,fny) 点をむすぶ
    LINE -(fnx(i), fny(i))
NEXT i
END
```

【結 果】



【応用問題 17】 無限大

無限大のマーク ∞ をつくるプログラムをつくれ。

【解 説】 ◎ 無限大のマークをつくります。

① $X=300+50\cos(x)+50\cos(x)$

$Y=200-50\sin(\cos(x)*\sin(x))$ とします。

② 無限大のマークとなります。

③ 4ヶ所 50 の値, 4ヶ所の x の値を変化させると無限の変形ができます。いろいろと試みてください。

【プログラム例】

```
SCREEN 0
CLS 0
DEF fnx (x) = 300 + 50 * COS(x) + 50 * COS(x) —— X 座標値
DEF fny (x) = 200 - 50 * SIN(COS(x) * SIN(x)) —— Y 座標値
PSET (fnx(0), fny(0)) —— i を 0 から  $2\pi$  まで
FOR i = 0 TO 2 * 3.14159 + .1 STEP .1 —— (fnx,fny) 点をむすぶ
    LINE -(fnx(i), fny(i))
NEXT i
END
```

【結 果】



【応用問題 18】 魚

【結 果】の魚のパターンをつくるプログラムをつくれ。

【解 説】 ◎ 魚のパターンをつくります。

① $X=300+100\cos(x)+100\cos(\cos^2(x))$

$Y=200-50\sin(\cos(x) \cdot \sin(x))$

として X を 0 から 2π ラジアンまでかきます。

② この他にもいろいろのパターンがつかれますので試みてください。

【プログラム例】

```
SCREEN 0
CLS 0
DEF fnx (x) = 300 + 100 * COS(x) + 100 * COS(COS(x) * COS(x)) —— X 座標値
DEF fny (x) = 200 - 50 * SIN(COS(x) * SIN(x)) —— Y 座標値
PSET (fnx(0), fny(0)) —— i を 0 から  $2\pi$  まで
FOR i = 0 TO 2 * 3.14159 + .1 STEP .1 —— (fnx,fny) 点をむすぶ
    LINE -(fnx(i), fny(i))
NEXT i
END
```

【結 果】



【応用問題 19】 アステロイド曲線

アステロイド曲線をかくプログラムをつくれ.

【解 説】 ①アステロイド曲線をかきます.

① $X=300+150\cos^3(x)$

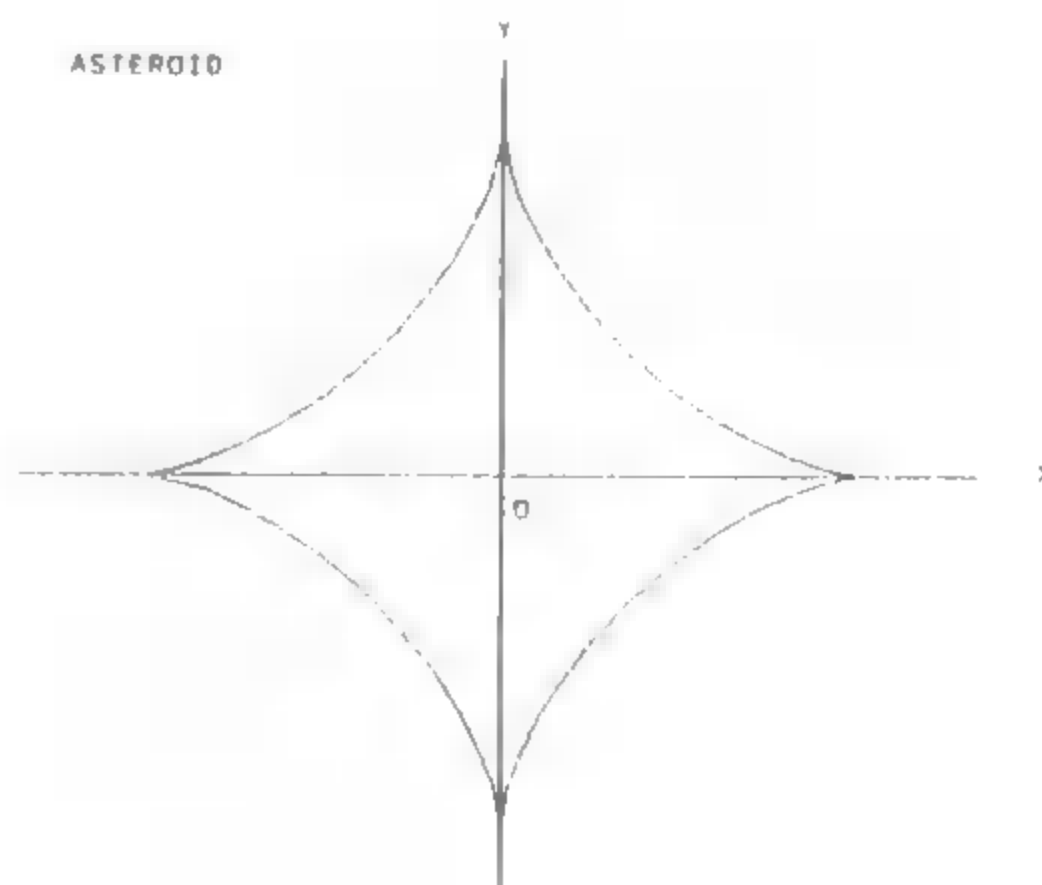
$Y=200-150\sin^3(x)$ とします.

② 【結 果】のアステロイド曲線となります.

【プログラム例】

```
SCREEN 0
CLS 0
DEF fnx (x) = 300 + 150 * COS(x) * COS(x) * COS(x)
DEF fny (x) = 200 - 150 * SIN(x) * SIN(x) * SIN(x)
LOCATE 2, 16: PRINT "ASTEROID"
LINE (300, 20)-(300, 380)
LINE (100, 200)-(500, 200)
LOCATE 1, 38: PRINT "Y"
LOCATE 14, 39: PRINT "O"
LOCATE 13, 66: PRINT "X"
PSET (fnx(0), fny(0))
FOR i = 0 TO 2 * 3.14159 + .1 STEP .1
    LINE -(fnx(i), fny(i))
NEXT i
END
```

【結 果】



【応用問題 20】 円の伸開線 (インボリュート)

円のインボリュートをかくプログラムをつくれ.

【解 説】 ◎ 円の伸開線（インボリュート）をかきます。

- ① インボリュートの X, Y 座標は次のようになります。

$$X=a(\cos(d)+d\sin(d))$$

$$Y=a(\sin(d)-d\cos(d))$$

- ② X, Y 座標を d を 0 ラジアンから $40*\pi$ ラジアンまで求めて結びます。

ここでは a を 1 とします。

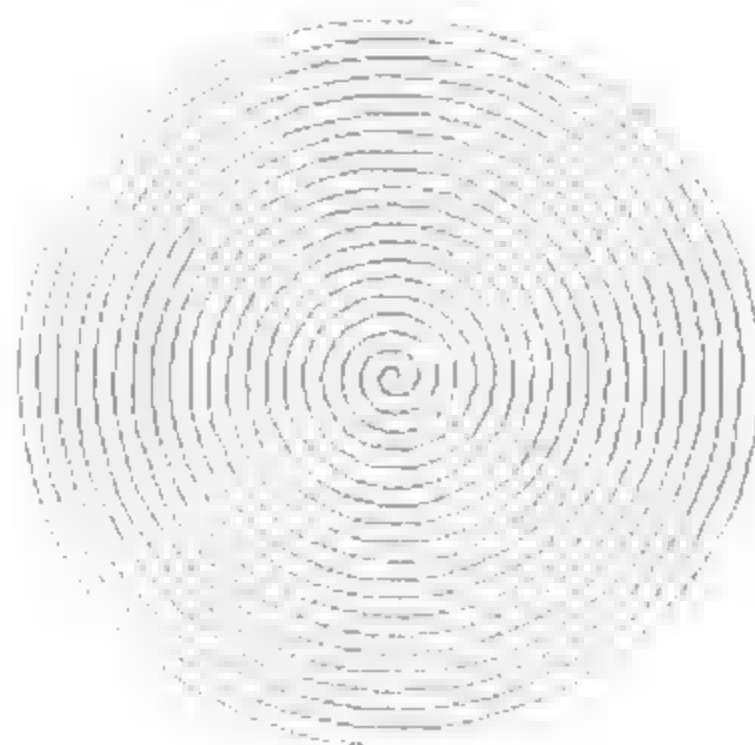
【プログラム例】

```
SCREEN 0
CLS 0
DEF fnx (d) = 300 + (COS(d) + d * SIN(d))
DEF fny (d) = 200 - (SIN(d) - d * COS(d))
PSET (fnx(0), fny(0))
  FOR i = 0 TO 40 * 3.14159 + .1 STEP .1
    LINE -(fnx(i), fny(i))
  NEXT i
END
```

X, Y 座標値の定義

表示

【結 果】



著 者 略 歴

田 中 廣 (たなか ひろし)

昭和14年 東京に生れる

昭和37年 慶応義塾大学経済学部卒

昭和37年 (株)安川電機製作所入社

昭和53年 安川商事(株)入社

著 書 ビジネスプログラミング・シリーズ

マイコンプログラミング500題

パソコンプログラミング500題

「PC-8801」周辺機器活用プログラミング300題

ポケットコンピュータプログラミング300題・シリーズ

フロッピーディスク活用入門

C言語プログラミング500題

PC-9801プログラミング500題

PC-9801機械語プログラミング500題

PC-9801活用プログラミング300題

PC-9801アイデアプログラミング300題

Lotus 1-2-3活用プログラミング300題

Multiplan 3.1活用プログラミング300題

dBASE III PLUS活用プログラミング300題

Quick Cプログラミング500題

書 名

Quick BASIC

プログラミング500題

NDC 007.64

(定価はカバーに表示してあります)

発行年月日

1989年11月30日 初版1刷発行

1993年7月7日 初版5刷発行

©著 者

田 中 廣

発 行 者

藤 吉 敏 生

発 行 所

日 刊 工 業 新 聞 社

東京都千代田区九段北一丁目8番10号

(〒102)

☎ 東京 3222-7111

振替口座 東京 9-186076

印刷・製本

大日本印刷株式会社

1989 Printed in Japan

乱丁・落丁本はお取り替えいたします。

ISBN4-526-02640-9 C3034

図 書 案 内

**C言語プログラミング500題
R U N C**

田中 廣 著
B 5 判 260頁 定価2200円

今後主流になるとみられるC言語プログラミングをマスターするためのテキスト、演習問題例示。

項目 文字変数 書式指定による出力 演算子 データの入力 算術関数 文字列操作関数 プリンタへの出力 配列のポインタ プリプロセッサ ほか

C 言 語 と U N I X

工博 松山泰男 著
A 5 判 340頁 定価2890円

UNIXのために開発されたC言語をUNIXとの関連で解説。“双方の上手なプログラミング手法がマスターできる”というのが狙いになっている。

項目 C言語によるプログラミング UNIXによるプログラミング環境 C言語とUNIXの概観 他

**PC-9801
パソコンプログラミング500題**

田中 廣 著
B 5 判 320頁 定価2060円

212の例題でBASIC言語の主要ステートメントの使い方を、演習問題258で学んだことの確認を、応用問題30題で応用展開のコツをつかめるよう構成。グラフィック、カラー、漢字、フロッピーディスク、ディジタイザ、プロッタ等の問題を多く出している。

**PC-9801
活用プログラミング300題**

田中 廣 著
B 5 判 296頁 定価2580円

パソコン500題シリーズの応用編。使用目的ごとにどのコマンドを使い、どの順序でプログラムを組むのかを、豊富なプログラム例で示すもの。

項目 基礎演算 表示 入力 プリント 文字データ くりかえし 配列 2次元グラフィックスほか

**PC-9801
機械語プログラミング500題**

田中 廣 著
B 5 判 368頁 定価2600円

PC-9801VM₂(CPUはV30)を使って、コマンドの使い方の例題を示しながら、機械語(アセンブラ)のプログラミング手法を解説した機械語入門書。

項目 入門編(機械語入門) 初級編(データ転送 他) 中級編(データ交換 他) 上級編 応用編

**Lotus 1-2-3
活用プログラミング300題**

田中 廣 著
B 5 判 350頁 定価2500円

統合ソフトとして評価の高い“Lotus 1-2-3”活用のための手引書。例題、練習問題、応用問題合わせて300題のプログラム例で、表計算などの手順を紹介。

項目 Lotus 1-2-3解説 ワークシート編(1)~(5) 算術関数編 日付関数・文字列関数編 グラフ編 他

**Multipan 3.1
活用プログラミング300題**

田中 廣 著
B 5 判 250頁 定価2500円

Multipan 3.1 の使い方を300題の例題をあげて、問題を解くことでマスターさせようとするもの。

項目 ワークシート編 算術関数編 日付関数編 文字列関数編 論理関数編 財務関数編 統計関数編 データベース編 グラフ編

「花子」活用プログラミング300題

田中 廣 著
B 5 判 364頁 定価2500円

人気の図形・グラフィックス用パソコンソフト「花子」を活用するための手引書。簡単な解説、入力方法、結果を解説する例題と練習問題等300題を例示。

項目 図形編 編集編 変更編 文字線 画面編 印刷・ファイル 補助線 グラフ編 応用編

**dBASE III PLUS
活用プログラミング300題**

田中 廣 著
B 5 判 400頁 定価2800円

dBASE III PLUSは米国アシュトンテートが開発したパソコン用リレーショナルデータベースのパッケージソフトである。本書は田中流にキー入力から応用までを300題の例題をあげて、段階的に実力を付けていくように編集されている。

実用 X-W i n d o w

辺見悦子 著
B 5 判 320頁 定価2900円

他のコンピュータからの情報を使用、活用するためのソフトとして注目されるWindowのうち、UNIX上で動くX-Windowについて、手法を解説。

項目 X-Window入門 X-Windowコマンドの利用 X-Windowのプログラミング X-Ray について

日刊工業新聞社

(上記定価には消費税3%が加えられておりません)

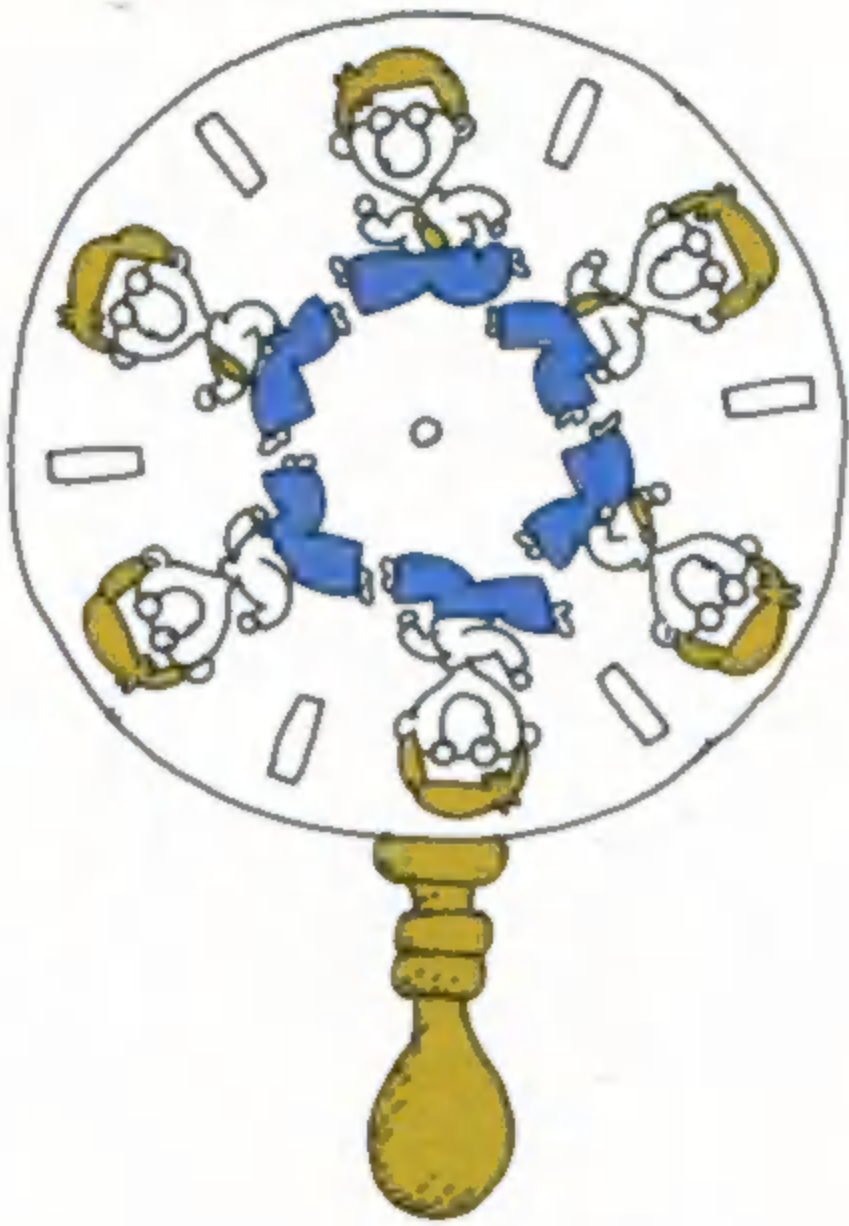
Quick BASIC

プログラミング500題

田中 廣 著



Quick BASIC



定価 3200円(本体 3107円)

ISBN4-526-02640-9 C3034 P3200E